ABSTRACT

Title of dissertation: TOWARDS GENERALIZED FRAMEWORKS FOR OBJECT RECOGNITION

Venkataraman Santhanam
Doctor of Philosophy, 2018

Dissertation directed by: Professor Larry S. Davis
Department of Computer Science

Over the past few years, deep convolutional neural network (DCNN) based approaches have been immensely successful in tackling a diverse range of object recognition problems. Popular DCNN architectures like deep residual networks (ResNets) are highly generic, not just for classification, but also for high level tasks like detection/tracking which rely on classification DCNNs as their backbone. The generality of DCNNs however doesn't extend to image-to-image(Im2Im) regression tasks (eg: super-resolution, denoising, rgb-to-depth, relighting, etc). For such tasks, DCNNs are often highly task-specific and require specific ancillary post-processing methods. The major issue plaguing the design of generic architectures for such tasks is the tradeoff between context/locality given a fixed computation/memory budget.

We first present a generic DCNN architecture for Im2Im regression that can be trained end-to-end without any further machinery. Our proposed architecture, the Recursively Branched Deconvolutional Network (RBDN), which features a cheap early multi-context image representation, an efficient recursive branching scheme

with extensive parameter sharing and learnable upsampling. We provide qualitative/quantitative results on 3 diverse tasks: relighting, denoising and colorization and show that our proposed RBDN architecture obtains comparable results to the state-of-the-art on each of these tasks when used off-the-shelf without any post processing or task-specific architectural modifications.

Second, we focus on gradient flow and optimization in ResNets. In particular, we theoretically analyze why pre-activation(v2) ResNets outperform the original ResNets(v1) on CIFAR datasets but not on ImageNet. Our analysis reveals that although v1-ResNets lack ensembling properties, they can have a higher effective depth in comparison to v2-ResNes. Subsequently, we show that downsampling projections (while only few in number) have a significantly detrimental effect on performance. We show that by simply replacing downsampling-projections with *identity-like dense-reshape* shortcuts, the classification results of standard residual architectures like ResNets, ResNeXts and SE-Nets improve by up to 1.2% on ImageNet, without any increase in computational complexity (FLOPs).

Finally, we present a robust non-parametric probabilistic ensemble method for *multi-classification*, which outperforms the state-of-the-art *ensemble methods* on several *machine learning* and *computer vision* datasets for *object recognition* with statistically significant improvements. The approach is particularly geared towards multi-classification problems with very low training data and/or a fairly high proportion of outliers, for which training end-to-end DCNNs is not very beneficial.

TOWARDS GENERALIZED FRAMEWORKS FOR OBJECT RECOGNITION

by

Venkataraman Santhanam

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2018

Advisory Committee:
Professor Larry S. Davis, Chair/Advisor
Professor Rama Chellappa, Dean's Representative
Professor Ramani Duraiswami
Professor David Jacobs
Professor Thomas Goldstein

Acknowledgments

I owe my sincere gratitude to everyone who has supported me this far.

First and foremost I'd like to thank my advisor, Professor Larry S. Davis for giving me complete autonomy to work on whatever crazy research idea I came up with and support me, not just financially, but also emotionally regardless of the outcome of my endeavors. This allowed me to focus purely on independent research without distractions from extraneous sources. It has been a pleasure to work with and learn from such an extraordinary individual.

I would also like to thank Dr. Vlad I. Morariu, who as a postdoc helped me immensely in my initial PhD stages. Dr. Vlad was always available to meet and would answer even the silliest of my doubts without disdain.

I owe my deepest gratitude to my family - amma and appa who have always stood by me and guided me through my career, and have pulled me through against impossible odds at times. Words cannot express the gratitude I owe them.

Finally, I would like to sincerely thank my friends, housemates and UMIACS colleagues for their friendship and support.

This research is based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via IARPA R&D Contract No. 2014-14071600012. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, or the U.S. Government. The U.S. Government is

authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

Thank you all!

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Given any image/video, it is often trivial for even small children to develop a basic understanding of the underlying scene. Humans have a potentially limitless capacity for remembering and identifying objects, which often makes *recognition* tasks seem fairly easy. Furthermore, humans also use context in a very natural way to develop complex relationships between various objects in a scene. For a machine however, such tasks are far from a trivial proposition. While we humans *can* perform these tasks at ease, we have yet to provide a thorough comprehensive description of precisely *how* we do them, despite decades of research in neurosciences. Any machine needed to automate even the most basic of tasks unfortunately doesn't have *intuition* or *common sense* at its disposal, and requires a precise set of instructions to either (a) directly perform the task or (b) learn from training data to develop a potentially complex set of instructions for performing the task.

Fully automated scene understanding has always been and is still considered the holy-grail of computer vision. While the problem remains unsolved at large, there has nevertheless been tremendous progress over the past few decades. A bulk of the progress has infact come over the last few years via *deep learning* approaches which have been made possible with major breakthroughs in hardware and compute

capabilities. Broadly speaking, deep learning relies on massive cascades of modular building blocks, each comprising of several basic compute units refered to as neurons (which are biologically inspired). A major reason for its widespread success is not just its extremely high modeling capacity, but the ability to learn *end-to-end* models for solving tasks. Deep Convolutional Neural Networks (DCNNs) in particular directly operate on the input image/video and learn to produce the desired output without relying on hand-crafted features or multi-step pipelines which are often ad-hoc/suboptimal.

1.1 The Importance of Image Classification

Ever since the advent of computer vision, the approach to solving complex high-level tasks has always been to break them heirarchically into progressively simpler low-level vision tasks. This approach works well since high level tasks are almost always strongly dependent on the output of one or more low-level tasks and can often be described as a repeated application of a low-level task. Figure 1.1 illustrates this for an example of traffic-jam analysis from video-footage. For analyzing such a video, the ability to *track* specific vehicles across video frames is often a pre-requisite. *Tracking* vehicles on the other hand requires the ability to reliably *detect* vehicles in individual frames. Finally, *detecting* vehicles involves applying a vehicle-classifier on various image-patches at various scales followed by non-maximal suppression. So, the vehicle *classification* task lies at the heart of this problem and is a critical component.

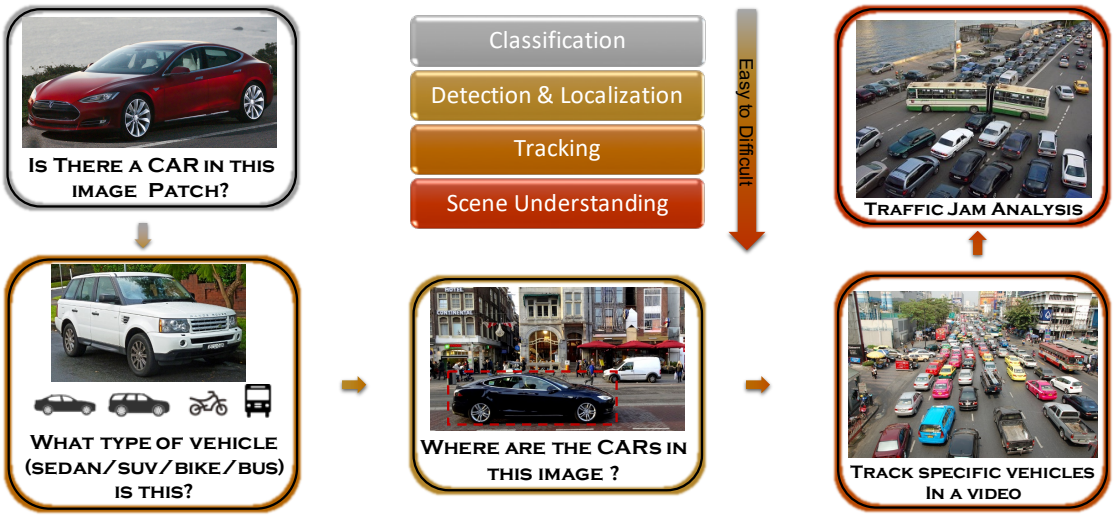

Figure 1.1: Illustrating Levels of Abstraction: Complex high-level problems in computer vision can often be hierarchically broken down into simpler low-level problems.

While this example might seem like a very specific case, a large number of high-level vision problems involve *object classification* in some stage or another. Improvements to the underlying classifier almost always translate to an improvement in the high-level task. For DCNNs in particular, historical improvements in ImageNet [6] classification, such as AlexNet [7] → VGG [8] → ResNet [1] → ResNeXt [9] have seen correspondingly major gains in object detection on Pascal-VOC [10] and MS-COCO [11].

DCNN classifiers pretrained on ImageNet can simply be finetuned to run as a detector with minimal modifications, by virtue of their stacked convolutional structure. While a DCNN classifier typically operates on a fixed-size input like 224×224 and provides a single classification score, the same classifier when applied on an arbitrary $H \times W$ sized image can be made to instead output a dense $H \times W$ grid of

classification scores by processing every overlapping 224×224 patch in the image.

1.2 Im2Im: Vision Tasks which do not rely on Image Classification

There are a wide class of vision tasks which however do not necessarily rely on image classification. A prominent class of such tasks is *Image-to-Image*(Im2Im) regression tasks, which take an image as input and produce another image as output. This involves tasks such as denoising, deblurring, jpeg-deblocking, super-resolution, style-transfer, rgb-to-depth, inpainting, relighting, colorization etc.

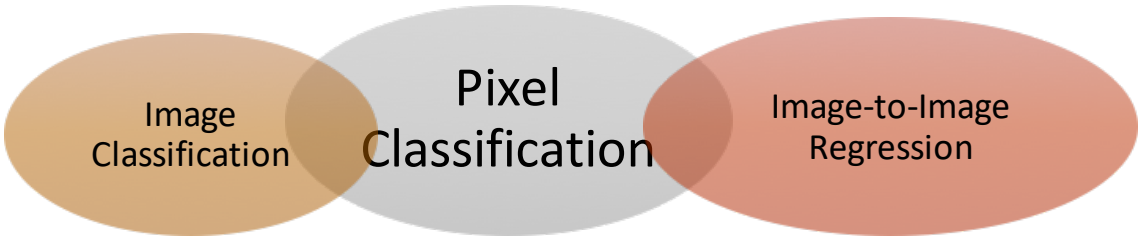

Figure 1.2: Not all vision tasks are dependent on image classification, most notably Im2Im regression tasks.

1.3 Limitations of DCNNs

DCNNs perform exceptionally well when ample training data is available. In cases where training data is scarce, DCNNs pretrained on ImageNet could still be used to finetune on the new data, as long as the new data has a reasonable domain overlap with one or more objects in ImageNet. However, in cases where both the data is scarce and the domain is significantly different from any other large

available dataset, DCNNs lose their efficacy. If the data additionally has outliers, the performance is furthur worsened. Medical data in particular often has these issues: data scarcity, domain divergence and potential outliers. Another limitation of DCNNs is their high computation and memory budget, often requiring expensive GPUs to run the networks.

1.4 Goals and Organization

In this thesis, we present novel approaches based on low-level vision, machine-learning and optimization theory, that can be used off-the-shelf to improve and augment a wide range of pre-existing object recognition pipelines. The thesis is organized as follows:

In chapter 2, we present a robust non-parametric probabilistic ensemble method for *multi-classification*, which outperforms the state-of-the-art *ensemble methods* on several *machine learning* and *computer vision* datasets for *object recognition* with statistically significant improvements. The approach is particularly geared towards multi-classification problems with very low training data and/or a fairly high proportion of outliers, for which training end-to-end DCNNs is not beneficial as described in 1.3.

In chapter 3, we focus our attention on image-to-image regression tasks. While generic DCNN classification architectures such as VGG [8],ResNet [1], *etc*.are highly generic and can be used for any classification task, there are currently no generic DCNN architectures for image-to-image regression. In particular, existing DCNNs

for Im2Im regression are often highly task-specific and require specific ancillary post-processing methods like CRFs. To that end, we present a generic DCNN for Im2Im regression which overcomes the limitations of its predecessors and can be used off-the-shelf to train arbitrary supervised Im2Im regression problems. Our network gives excellent results on a wide range of Im2Im regression tasks such as denoising, relighting and colorization. For denoising in particular, we obtain state-of-the-art results using a single model for blind-denoising which even beats noise-level specific competitor approaches at high noise levels by ~1db.

In chapter 4, we focus on optimization and gradient flow in deep residual networks and compare/contrast its two proposed variants [1, 2]. Pre-activation ResNets [2] consistently outperform the original post-activation ResNets [1] on the CIFAR10/100 classification benchmark. However, these results surprisingly don't carry over to the standard ImageNet benchmark. We first theoretically analyze this incongruity in terms of how the 2 variants differ in handling the propagation of gradients. While identity shortcuts are critical in both variants for improving optimization and performance, we show that post-activation variants enable early layers to receive a diverse dynamic composition of gradients from effectively deeper paths in comparison to pre-activation variants, enabling the network to make maximal use of its representational capacity. Secondly, we show that downsampling projections (while only few in number) have a significantly detrimental effect on performance. We show that by simply replacing downsampling-projections with *identity-like dense-reshape* shortcuts, the classification results of standard residual architectures like ResNets [1], ResNeXts [9] and SE-Nets [12] improve by up to 1.2%

on ImageNet, without any increase in computational complexity (FLOPs).

Chapter 2: KDEMRP: A Robust Non-Parametric Ensemble Method

2.1 Overview

In this chapter, we focus on a generic approach for solving *multi-classification* problems using *ensemble methods*. This approach involves dividing the *multi-classification* problem into a set of binary classification problems and using an *ensemble method* to combine the binary classifier scores into a multi-classification output. Binary classifiers are often easier to build, faster to train/test and have much simpler decision boundaries when compared to dedicated multi-class classifiers. Moreover, for complex multi-class problems with a large number of classes and/or high feature dimensionality, it is often practically more viable to divide the multi-classification problem into several smaller easy-to-solve binary classification problems.

Several binary decomposition strategies exist for the multi-class problem, with the most popular ones being the *one-vs-all* (*OVA* [13]) and *one-vs-one* (*OVO* [14]) schemes. In *OVA*, a binary classifier is trained for each class, designed to distinguish it from the remaining classes. *OVO* on the other hand, trains a binary classifier to distinguish between every pair of classes. Yet another decomposition strategy, that can be viewed as a generalization of *OVO* and *OVA*, is the *error-correcting-output-code* (*ECOC* [15]) framework, in which each class is assigned a unique fixed length

binary codeword, after which a binary classifier for each bit position is trained, based on the codewords for all the classes. Minimal design *ECOCs* offering competitive performance have also been proposed [16], for which the number of binary classifiers required is sub-linear in the number of classes.

Rifkin and Klautau argued that *OVA* can match *OVO* performance, provided the binary classifiers are well tuned [17]. Their analysis is however restricted to regularized classifiers such as *SVMs*, and is not applicable for generic binary classifiers. The *OVO* scheme typically provides better results than the *OVA* or *ECOC* scheme [18, 19]. *OVO* is also more robust over various choices of binary classifiers and provides better scalability with a relative performance boost over *OVA* as the number of classes increases [20]. *OVO* is also surprisingly faster to train than *OVA* (and sometimes even *ECOC*), despite training more binary classifiers. This is because each *OVO* binary classifier is trained only on samples from a specific pair of classes, whereas each binary classifier in *OVA* or *ECOC* is trained using samples from all classes. Furthermore, when parallel computing is available, all *OVO* pairwise classifiers can be trained in a massively parallel fashion, even for a very high number of classes.

After a binary decomposition of the multi-classification problem, the resulting binary classifier scores are aggregated to yield a final multi-classification output by strategies that are referred to as *ensemble methods* [21]. Our contribution is a new *OVO ensemble* method using *KDE* over *PCA projections* of binary classifier scores that is robust, yields probabilistic multi-class outputs and outperforms the most commonly used alternatives *VOTE/NEST*.

The rest of the chapter is organized as follows : Section 2.2 details related work in *OVO ensemble methods*. Section 2.3 introduces notations, formalizes the notion of an ensemble method and describes our proposed approach. Section 2.4 is dedicated to *Kernel Density Estimation(KDE)*, along with a *PCA projection* based approximation to multi-variate *KDE* which we use to obtain our probabilistic multi-class decisions. Sections 2.5, 2.6 describe the experiments to test our approach against the state-of-the-art and contain a discussion of the results. Finally, we conclude in section 2.7 with a summary of our proposed approach, its novel contributions and potential future improvements.

2.2 Related Work

The most common *OVO ensemble method* is the *naïve voting scheme* (*VOTE* [22]), where all pairwise binary classifiers vote for the class of an unseen sample. The class with the highest number of votes is chosen as the predicted class. An improvement to *VOTE* is the *nesting one-vs-one scheme* (*NEST* [23]), which augments *VOTE* with a recursive tie-breaking scheme for instances where there are ties for the class with the highest vote. Several other *ensemble methods* have been proposed for *OVO* schemes, such as *weighted voting* (*WV* [24]), *Pairwise Coupling* (*PC* [25,26]), *decision directed acyclic graph* (*DDAG* [27]), *learning valued preference for classification* (*LVPC* [28]), *preference relations solved by non-dominance criterion* (*ND* [29]) and *binary tree of classifiers* (*BTC* [30]).

An excellent overview and a detailed experimental study, of these *OVO en-*

semble methods for various choices of binary classifiers is provided in [20]. Their results indicate that there is no "one method which performs best for all binary classifiers." The methods which perform consistently well, regardless of the choice of binary classifiers, are in fact the ones that are the simplest to explain: *VOTE* and *NEST*. Most of the popular machine learning software libraries used extensively by researchers, such as *LibSVM*, use *VOTE* as their *ensemble* method.

2.2.1 Limitations of *VOTE, NEST*

For a multi-classification problem with K classes, the *OVO* scheme trains $K(K\text{-}1)/2$ binary classifiers to distinguish between each pair of classes. For a test sample belonging to a class A, there will be $(K\text{-}1)(K\text{-}2)/2$ classifiers that have never seen any sample from class A. The predictions of any of these classifiers for the sample becomes arguably questionable. This is a recurring issue for all *OVO* ensemble methods, often referred to as the *non-competence problem* [31].

Considering the fact that both the *VOTE* and *NEST* methods disregard the relative magnitudes of the classifier scores completely and focus only on the binary predictions (votes), the vote given by a *non-competent* classifier is given the same weight as the vote carried by a *competent* one, which may affect results negatively. The success of the *VOTE* or *NEST* method, despite this limitation, is justified by the inherent redundancy in the *OVO* framework, with the rationale that the $(K\text{-}1)$ *competent* classifier votes more than compensate for the apparently random votes of the *non-competent classifiers*, which are usually not directed in favour of any one

particular class.

However, there may be cases where this assumption is violated, especially for samples that tend to be confused between a pair or small subset of classes. Such samples tend to have several classes with a comparable (albeit low) vote count. The votes of *non-competent classifiers*, in this case tend to have a stronger influence on the final prediction.

In order to overcome this limitation, a mechanism needs to be devised to efficiently utilize the information contained in the magnitudes of the scores. However, the raw scores for different *OVO* classifiers are not calibrated, centered at different thresholds, and have a potentially different scale and range. So, it would be unwise to use raw score magnitudes in any multi-classification scheme. The most intuitive way to make the scores of different *OVO* classifiers comparable is to convert them to probabilities.

2.2.2 Addressing the Non-Competence Problem

A couple of approaches, *DCS* [32] and *DRCW-OVO* [33] have been recently proposed which specifically aim to reduce the *non-competence* effect. Both these techniques focus on *removal(DCS)* or *reweighting(DRCW-OVO)* of *non-competent* classifiers and can in fact be used as a pre-processing step for any subsequent *OVO* ensemble method (in place of *weighted voting* used in *DCS* and *DRCW*).

In *DCS(Dynamic Classifier Selection)*, $3K$ nearest neighbors in the training data are first computed for each test sample based on the original feature space. An

i-vs-j OVO classifier is flagged as *non-competent* (and removed) with respect to a given test sample, if its $3K$ neighbors contain no samples from either of class i or class j.

In *DRCW-OVO (Distance-based Relative Competence Weighting)*, distances $(d_1, ..., d_K)$ are computed for each test sample, where d_i is the average distance to the 5 nearest neighbors from class i training samples in the original feature space. Subsequently, the score s_{ij} of each *i-vs-j OVO classifier* is reweighted as $s^*_{ij} = s_{ij} * d_j^2/(d_i^2 + d_j^2)$. The rationale behind this transformation is that the scores for the *competent* classifiers will be skewed more in favour of the true class whereas the scores of the *non-competent* classifiers will not be affected as much (due to the ratio $d_j^2/(d_i^2 + d_j^2)$ being closer to 0.5).

2.2.3 Probability Estimates for Binary Classification

The sign of a binary classifier score represents the predicted class, while its magnitude crudely encodes the confidence of the predicted decision. Several *classifier specific* methods that convert scores to probabilities exist. For max-margin classifiers, such as *Linear SVM*, techniques such as *Platt scaling* convert *scores to probabilities* by means of a *sigmoidal* function [34,35]. On the other hand, for classifiers such as *Naïve Bayes*, *Platt scaling* performs poorly while *isotonic regression* [36] gives better probability estimates [37]. It is also shown in [37], that neither *Platt scaling* nor *isotonic regression* perform well for classifiers such as neural nets, bagged trees, and logistic regression which already provide well calibrated scores.

2.2.4 Probability Estimates for Multi-classification

Obtaining multi-class probability estimates is a much trickier proposition. For the *OVO* scheme, we are required to estimate a probability for each class given all the pairwise binary classifier scores. Previously proposed methods typically employ a two-step strategy: convert each binary classifier score to a probability r_{ij}, and then combine r_{ij}'s to obtain the multi-class probabilities p_i's. The combination strategy is formulated as an optimization problem that attempts to bridge the gap between ratios p_i/p_j and r_{ij}/r_{ji}, with the constraint that $\sum p_i = 1$. The various methods mainly differ in how they formulate and solve this optimization problem, with techniques such as the least-squares method with non-negativity constraints [38], pairwise coupling with minimized *Kullback Leibler* divergence [25] and pairwise coupling with minimized log-loss [26].

The key limitation of these two-step approaches is that good quality estimates r_{ij}'s are crucial to the success of the method. Estimation errors in r_{ij}'s get compounded when estimating p_i's. Our proposed *ensemble* approach is non-parametric, makes no classifier-specific assumptions and directly estimates multi-class probabilities p_i's as joint densities over the pairwise classifier scores.

2.3 Ensemble Method Formalization

We begin by introducing notation and developing a formulation for a generic *OVO ensemble* method. We represent the *VOTE* scheme within our *ensemble* formulation in section 2.3.1. In section 2.3.2, we provide a high level overview of

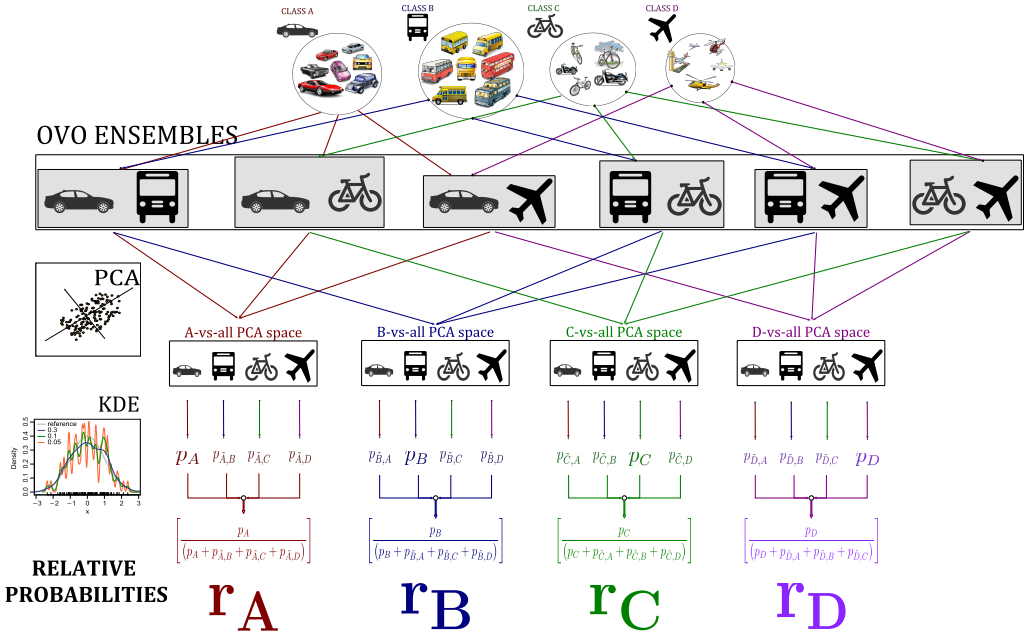

Figure 2.1: High level overview of the pipeline for our proposed **KDEMRP** *OVO ensemble* method, for a 4 class toy example, in a top-down fashion. The first step is the training of all the 6 pairwise *OVO ensembles*. Then for each class, *PCA* is used on a selected set of scores of all the training data. Subsequently, *KDE* is applied to estimate the various densities after which the final classification is done via *Max-Relative-Probability*

our proposed approach. In section 2.3.3, we introduce a partitioning over the set of all pairwise classifier scores, which isolates the influence of competent and non-competent classifiers for all classes. We make use of this partitioning by estimating the density of each class under a different partition of scores.

For a K-class multi-classification problem, with classes indexed by $\mathcal{I}=\{1,\ldots,K\}$, we are typically provided finite training data $\boldsymbol{T}_c=\{\boldsymbol{t}_{c_k}\}\ \forall c\in\mathcal{I}$, where $|\boldsymbol{T}_c|=n_c\in\mathbb{N}$ and the training samples $\boldsymbol{t}_{c_k}\in\mathbb{R}^d$ come from a potentially high d-dimensional feature space. Let $\boldsymbol{\mathcal{T}}=\bigcup_{c\in\mathcal{I}}\boldsymbol{T}_c$ denote all the training data. Let $D=K(K\text{-}1)/2$ denote the total number of pairs of classes.

Given a binary classifier f, the *OVO* scheme computes a family of D pairwise classifier models $\mathcal{F}=\{f_{ij}\}_{i>j}^{i,j\in\mathcal{I}}$ using the training data $\boldsymbol{\mathcal{T}}$. Each classifier model $f_{ij}:\mathbb{R}^d\rightarrow\mathbb{R}$ is a mapping from the feature space to a scalar classifier score and can be interpreted as a binary classifier f trained to discriminate between classes i and j. Model f_{ij} classifies a test sample $\boldsymbol{x}\in\mathbb{R}^d$ as class i if the score $f_{ij}(\boldsymbol{x})\geq 0$ and class j otherwise.

The Cartesian product of all mappings within the family $\mathcal{F}$, $\bigtimes_{f\in\mathcal{F}} f$ induces a conglomerate mapping $\boldsymbol{\mathcal{M}}:\mathbb{R}^d\rightarrow\mathbb{R}^D$, which maps any sample $\boldsymbol{x}\in\mathbb{R}^d$ in the feature space to a score vector $\boldsymbol{s}\in\mathbb{R}^D$,

$$\boldsymbol{s}=(s^{2,1},s^{3,1},s^{3,2},\ldots,s^{K,1},\ldots,s^{K,K-1}),$$

where $s^{i,j}=f_{ij}(\boldsymbol{x})$. For the sake of convenience, we additionally define $s^{j,i}=-s^{i,j}$ and $s^{i,i}=0\ \forall i,j\in\mathcal{I}$, which enables us to represent $\boldsymbol{s}$ as a $K*K$ *skew-symmetric* score matrix $\boldsymbol{S}$ with elements $s^{i,j}$. From here on, we will use both representations $\boldsymbol{s}$ and

$\boldsymbol{S}$ interchangeably to denote the D scores. Let $\boldsymbol{\mathcal{M}}(\boldsymbol{T}_c)=\{\boldsymbol{\mathcal{M}}(\boldsymbol{t}_{c_k})\}=\{\boldsymbol{s}_{c_k}\}$. Then $\boldsymbol{\mathcal{M}}(\boldsymbol{\mathcal{T}})$ represents the set of classifier scores of all the training data.

A non-probabilistic *OVO ensemble* method $\mathcal{E}_{\boldsymbol{\mathcal{M}}(\boldsymbol{\mathcal{T}})} : \mathbb{R}^D \rightarrow \mathcal{I}$, for a given set of training data scores $\boldsymbol{\mathcal{M}}(\boldsymbol{\mathcal{T}})$, takes as input the score vector $\boldsymbol{s}=\boldsymbol{\mathcal{M}}(\boldsymbol{x})$ of a test sample $\boldsymbol{x}\in\mathbb{R}^d$ and outputs the predicted multi-class decision $c\in\mathcal{I}$. This definition implies that the original multi-classification problem with training data $\boldsymbol{\mathcal{T}}$ on $\mathbb{R}^d$ is first transformed into an alternate multi-classification problem on $\mathbb{R}^D$, which is then solved by the *OVO ensemble* method using training data $\boldsymbol{\mathcal{M}}(\boldsymbol{\mathcal{T}})$.

A probabilistic *OVO ensemble* method $\boldsymbol{\mathcal{E}}_{\boldsymbol{\mathcal{M}}(\boldsymbol{\mathcal{T}})} : \mathbb{R}^D \rightarrow [0,1]^K$ provides a K-*tuple* of probability (confidence) estimates for the test sample respectively belonging to each of the K classes:

$$\boldsymbol{\mathcal{E}}_{\boldsymbol{\mathcal{M}}(\boldsymbol{\mathcal{T}})}(\boldsymbol{s}) = \langle \hat{p}_1(\boldsymbol{s}), \ldots, \hat{p}_K(\boldsymbol{s}) \rangle \,,$$

where $\hat{p}_c(\boldsymbol{s})$ is the estimated probability that the test sample belongs to class c, based on its scores $\boldsymbol{s}$.

Our goal is to design a probabilistic *OVO ensemble* method that uses $\boldsymbol{\mathcal{M}}(\boldsymbol{\mathcal{T}})$ to obtain the multi-class probability estimates from the D scores, without making any assumptions on the binary classifier f.

2.3.1 The *VOTE* scheme

VOTE has a rather simple representation in our non-probabilistic ensemble formulation:

$$VOTE_{\boldsymbol{\mathcal{M}}(\boldsymbol{\mathcal{T}})}(\boldsymbol{s}) = \underset{c}{\operatorname{argmax}} \left(\sum_{i\in\mathcal{I}} \operatorname{sgn}(s^{c,i}) \right) ,$$

where sgn is the *sign* function. *VOTE* does not utilize $\boldsymbol{\mathcal{M}}(\boldsymbol{\mathcal{T}})$ in any way and bases its decision solely on the *signs* of scores $\boldsymbol{s}$ for the test sample. It is important to note that, when given access to *only* the scores $\boldsymbol{s}$ (without access to $\boldsymbol{\mathcal{M}}(\boldsymbol{\mathcal{T}})$), it is hard to justify using anything other that *VOTE* without making strong assumptions on the classifier f. The score magnitudes $|s^{i,j}|$ for different i, j are across potentially different scales and mutually incomparable when no *apriori* knowledge is available regarding the distribution of the scores.

2.3.2 Proposed Approach: KDEMRP

We provide a high level overview of our proposed probabilistic ensemble method here. With the availability of $\boldsymbol{\mathcal{M}}(\boldsymbol{\mathcal{T}})$, we can obtain *data-driven* probability estimates from the scores. Non-parametric density estimation techniques such as *KDE* are an excellent choice for converting univariate scores to probabilities. However, for multi-variate data, *KDE* is expensive and requires a lot of data, which is often not available. We introduce a univariate approximation to multi-variate *KDE* based on *PCA* projections in section 2.4.2, which we use to estimate our joint densities from multi-variate scores in our proposed approach.

We start by using *KDE* on $\boldsymbol{\mathcal{M}}(\boldsymbol{\mathcal{T}})$ to estimate densities $\hat{g}_c(\boldsymbol{s}), \hat{g}_{\bar{c}}(\boldsymbol{s})\ \forall c \in \mathcal{I}$ from the scores, where $\hat{g}_c(\boldsymbol{s})$ is the estimated density of class c samples (via *KDE* on $\boldsymbol{\mathcal{M}}(\boldsymbol{T}_c)$) and $\hat{g}_{\bar{c}}(\boldsymbol{s})$ is the estimated density of all samples not belonging to class c (via *KDE* on $\boldsymbol{\mathcal{M}}(\boldsymbol{\mathcal{T}}/\boldsymbol{T}_c)$).

We then compute relative probabilities

$$\hat{r}_c(\boldsymbol{s}) = \frac{\hat{g}_c(\boldsymbol{s})}{\hat{g}_c(\boldsymbol{s}) + \hat{g}_{\bar{c}}(\boldsymbol{s})},$$

for which the decision rule $c^* = \text{argmax}_c\, \hat{r}_c(s)$ can be used to obtain the predicted multi-classification decision. Finally, the relative probabilities can be normalized to obtain the absolute ensemble probability estimates $\hat{p}_c(\boldsymbol{s}) = \hat{r}_c(\boldsymbol{s}) / \sum_{c' \in \mathcal{I}} \hat{r}'_c(\boldsymbol{s})$.

We call this *KDE+ Max Relative Probability* based approach *KDEMRP* from here on. We describe the details behind the estimation of the joint densities $\hat{g}_c(\boldsymbol{s}), \hat{g}_{\bar{c}}(\boldsymbol{s})$ in section 2.4.3.

2.3.3 Partitioning the classifier score space $\mathbb{R}^D$

For a test sample with scores $\boldsymbol{s}$, any row c of the score matrix $\boldsymbol{S}$ excluding the diagonal element, has the following K-1 components: $\left(s^{c,1}, \ldots, s^{c,c-1}, s^{c,c+1}, \ldots, s^{c,K}\right)$. These are the scores for classifier models which discriminate c from each of the remaining K-1 classes. If the test sample is indeed from class c, it is precisely these scores which correspond to the *competent* classifiers (mentioned in 2.2.1) , while the remaining $(K\text{-}1)(K\text{-}2)/2$ scores come from *non-competent* classifiers which have never seen any sample from class c.

We partition our space of classifier scores $\mathbb{R}^D$ into K groups of $\mathbb{R}^{K\text{-}1}$ corresponding to each of the K classes. For any class c, the components $\left(s^{c,1}, \ldots, s^{c,c-1}, s^{c,c+1}, \ldots, s^{c,K}\right)$ of the scores comprise its associated $\mathbb{R}^{K\text{-}1}$ space, which we call the *c-vs-all* space.

To reduce the influence of *non-competent* classifiers, we estimate each density

pair $\hat{g}_c(\boldsymbol{s}), \hat{g}_{\bar{c}}(\boldsymbol{s})$ on its separate *c-vs-all* space.

2.4 Kernel Density Estimation

Kernel Density Estimation (KDE) is one of the oldest non-parametric density estimation techniques (possibly preceded only by the histogram). We first describe univariate *KDE* in section 2.4.1. In section 2.4.2, we describe the iterative univariate approximation to multi-variate *KDE* over *PCA* space. In section 2.4.3, we describe the estimation of the joint densities used in our proposed *KDEMRP* approach: $\hat{g}_c(\boldsymbol{s})$, $\hat{g}_{\bar{c}}(\boldsymbol{s})$ in detail.

2.4.1 Univariate KDE

For univariate density estimation, with n scalar training data points $\{x_1, \ldots, x_n\}$, the *KDE* process involves the summation of a kernel function $\mathcal{K}$ (symmetric function which integrates to 1) centered at each of the n data points. The resulting summation, after normalization becomes the *estimated density function* $\hat{g}()$:

$$\hat{g}(x) = \frac{1}{nh} \sum_{i=1}^{n} \mathcal{K}\left(\frac{x - x_i}{h}\right)$$

The smoothing parameter h is called the *bandwidth*, the choice of which is critical in order to get a good density estimate. Significant research exists on developing better *objective data-driven bandwidth selection* methods for KDE ([39] provides an excellent survey). The *Gaussian* kernel $\mathcal{K}(x)=(2\pi)^{-1/2}e^{-x^2/2}$ is the most widely studied and is usually the first choice in the absense of prior knowledge about the potential *distribution* or *modalities*, which our data possesses. The most

popular choice of bandwidth for the *Gaussian* kernel is given by *Silverman's rule of thumb* [40] : $h \approx 1.06\sigma n^{-1/5}$, where σ, n are respectively the *sample standard deviation, size* of the training data.

Although *KDE* is efficiently computable for univariate distributions, multi-variate *KDE* is often shunned by researchers, since not only is there an increase in the computational cost, but also the number of training points required to get a reliable density estimate increases exponentially with data dimensionality.

2.4.2 KDE over PCA space

We use an approximation to the prohibitively expensive multi-variate *KDE* that involves projecting the multi-variate data on to *PCA* spaces and independently applying univariate *KDE* along each of the *PCA* projections. For *PCA* with l factors, we approximate the joint density as

$$\hat{g}(x_1, \cdots, x_l) \approx \prod_{i=1}^{l} \hat{g}(x_i).$$

Since PCA directions are selected in the decreasing order of data-variance, the approximation essentially assumes that the major sources of variation for the data are independent. For our multi-classification task, we make this assumption on selected groups of pairwise classifier scores, exploiting the work already done by the pairwise binary classifiers in our *OVO ensemble*. These assumptions are not classifier-specific, but rather rely on the overall good discriminatory ability of each pairwise classifier.

It is also worthwhile to note that even if we had the luxury of time to directly do multi-variate *KDE*, we typically never have enough data to get reliable multi-variate density estimates.

2.4.3 Estimation of joint densities

We finally describe the estimation of the joint densities $\hat{g}_c(\boldsymbol{s})$, $\hat{g}_{\tilde{c}}(\boldsymbol{s})$, which appear in the expression for relative probabilities in our proposed *KDEMRP* approach.

We apply *PCA* on the components of $\boldsymbol{\mathcal{M}}(\boldsymbol{\mathcal{T}})$ on the *c-vs-all* space, and call the resulting space the *c-vs-all PCA* space, which we formalize by defining projections

$$\boldsymbol{\mathcal{P}}_c : \mathbb{R}^D \rightarrow \mathbb{R}^{K-1} \quad \forall c \in \mathcal{I},$$

where $\boldsymbol{\mathcal{P}}_c$ takes as input a score $\boldsymbol{s}$ and maps its components $\left(s^{c,1}, \ldots, s^{c,c-1}, s^{c,c+1}, \ldots, s^{c,K}\right)$ to the *c-vs-all PCA* space $\boldsymbol{\mathcal{P}}_c(\boldsymbol{s}) = \boldsymbol{x}^c = (x_1^c, \ldots, x_{K-1}^c)$. For any class $c \in \mathcal{I}$, let $\boldsymbol{\mathcal{P}}_c\left(\boldsymbol{\mathcal{M}}(\boldsymbol{T}_c)\right)$ denote the projections of the scores of class c training samples on to the *c-vs-all PCA* space, and let $\boldsymbol{\mathcal{P}}_c\left(\boldsymbol{\mathcal{M}}(\boldsymbol{\mathcal{T}}/\boldsymbol{T}_c)\right)$ denote the projections of the scores of all other training samples (excluding class c) on to the *c-vs-all PCA* space.

We estimate densities $\hat{g}_c(\boldsymbol{s})$ $\forall c \in \mathcal{I}$ as

$$\hat{g}_c(\boldsymbol{s}) = \hat{g}_c\left(\boldsymbol{\mathcal{P}}_c(\boldsymbol{s})\right) = \prod_{i=1}^{K-1} \hat{g}_c(x_i^c) \Bigg|_{\substack{\text{KDE over} \\ \boldsymbol{\mathcal{P}}_c(\boldsymbol{\mathcal{M}}(\boldsymbol{T}_c))}},$$

where $\hat{g}_c(\boldsymbol{s})$ represents the estimated density of class c samples in the *c-vs-all PCA* space. We estimate densities $\hat{g}_{\tilde{c}}(\boldsymbol{s})$

$$\hat{g}_{\tilde{c}}(\boldsymbol{s}) = \hat{g}_{\tilde{c}}\left(\boldsymbol{\mathcal{P}}_c(\boldsymbol{s})\right) = \prod_{i=1}^{K-1} \hat{g}_{\tilde{c}}(x_i^c) \Bigg|_{\substack{\text{KDE over} \\ \boldsymbol{\mathcal{P}}_c(\boldsymbol{\mathcal{M}}(\boldsymbol{\mathcal{T}}/\boldsymbol{T}_c))}},$$

where $\hat{g}_{\bar{c}}(\boldsymbol{s})$ represents the estimated density of all samples not belonging to class c in the *c-vs-all PCA* space.

2.5 Experiments on KEEL Multi-classification Datasets

We first focus on general multi-classification and compare our approach against *state-of-the-art OVO* ensemble methods *DCS* and *DRCW*. Our experiments are carried out on datasets from the *KEEL* repository [41], using a very similar experimental setup as in [33]. Subsequently, we perform statistical tests to see if significant differences exist between approaches.

2.5.1 Datasets

We use 16 datasets from the *KEEL* repository for our experiments. The datasets *car, lymphography, nursery, dermatology, flare, led7digit* have a mixture of nominal and numerical attributes, while the rest have only numerical attributes.

Some of these datasets have a few classes with an extremely small number of samples, making density estimation infeasible for those classes (Table 2.1 shows the datasets used and the number of samples per class). In order to solve this problem, we first isolate all classes with less then 40 samples in the training data. Subsequently we remove all samples belonging to these classes from the training data and also remove all the rows and columns corresponding to these classes from the confidence matrix of each sample. Then, the *KDEMRP* densities are estimated only to discriminate amongst the remainder of the classes. During test time, a

Data-set	**#Ex.**	**#Cl.**	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}
Car	1728	4	1210	384	65	69							
Lymphography	148	4	2	81	61	4							
Vehicle	846	4	199	217	218	212							
Cleveland	297	5	160	54	35	35	13						
Nursery	1296	5	1	32	405	426	432						
Shuttle	2175	5	1706	2	6	338	123						
Dermatology	358	6	111	60	71	48	48	20					
Flare	1066	6	331	239	211	147	95	43					
Glass	214	7	70	76	17	0	13	9	29				
Satimage	643	7	154	70	136	62	71	0	150				
Segment	2310	7	330	330	330	330	330	330	330				
Ecoli	336	8	143	77	2	2	35	20	5	52			
Led7digit	500	10	45	37	51	57	52	52	47	57	53	49	
Penbased	1100	10	115	114	114	106	114	106	105	115	105	106	
Yeast	1484	10	244	429	463	44	51	163	35	30	20	5	
Vowel	990	11	90	90	90	90	90	90	90	90	90	90	90

Table 2.1: Number of samples per class for the 16 KEEL datasets used in the experiments.

sample is first classified by *DRCW*. If the prediction belongs to one of the small sample classes, then the *DRCW* prediction is retained. If not, then *KDEMRP* is used instead to classify the sample.

3 additional datasets: *zoo, pageblocks* and *autos* were used in [33]. We decided to drop those, since they contain too few samples in all classes for any reliable density estimation. In particular, for each of these 3 datasets, the class with the second highest number of samples had fewer than 40 training samples per class.

2.5.2 Base Classifier

We use *PDFC (Positive Definite Fuzzy Classifier)* as the *base classifier*, since *PDFC* was shown in [33] to give the best average performance for these datasets over other *base classifiers* such as *C45, Ripper, kNN* and *SVM*. We use the same parameters for *PDFC* as in [33].

PDFC however only provides binary outputs (1/0) rather than a continuous spectrum of scores which we prefer for good density estimation. We deal with this by giving *KDEMRP* the *DRCW-reweighted* confidence matrix as input instead of the original confidence matrix. As a consequence, we do not directly compare between *DRCW* and *KDEMRP*, but instead see if *KDEMRP* applied on top of the *DRCW* confidences can improve the results of *DRCW* alone. We denote the combination *(DRCW+KDEMRP)* as KDEMRP† from here on.

2.5.3 Experimental Framework

Apart from *DRCW*, we also use *DCS* and *PC (Pairwise Coupling)* for comparison(*PC* was shown [32] to give the best results for these datasets over other methods like *ND, WV* and *PE* when *PDFC* was the base classifier).

For performance measures, we use the accuracy rate and *Cohen's kappa* [42] as in [33].

We then perform the non-parametric *Friedman aligned-ranks* test [43] like in [33] followed by the *Holm post-hoc* test [44], separately for both measures accuracy and kappa. The *Friedman aligned-ranks* test first detects significant difference among the approaches by ranking them based on the performance metric and picks the best approach (lowest rank) which is referred to as the *control* algorithm. The *Holm post-hoc* test then generates a *p-value* which indicates whether the *control* algorithm is significantly better than the others.

Finally, we directly compare *DRCW* with KDEMRP† using the *Wilcoxon signed-ranks* test [45].

2.5.4 Results

Table 2.2 shows the average 5-fold test accuracy and kappa values for *PC, DCS, DRCW* and KDEMRP†. The best results for each dataset are stressed in boldface. At a cursory glance, we can see that KDEMRP† does give the best results for the majority of datasets. It also has the highest average accuracy and kappa over all datasets.

Data-set	Average Accuracy				Average Kappa			
	PC	DCS	DRCW	KDEMRP†	PC	DCS	DRCW	KDEMRP†
Car	99.77	**99.88**	99.42	99.77	0.9950	**0.9975**	0.9874	0.9950
Cleveland	53.92	55.93	56.61	**56.93**	0.2818	0.2888	0.3088	**0.3129**
Dermatology	84.66	**93.85**	91.90	89.39	0.8011	**0.9219**	0.8964	0.8638
Ecoli	84.07	83.78	**84.68**	**84.68**	0.7801	0.7751	0.7878	**0.7886**
Flare	73.64	**73.92**	73.69	73.88	0.6594	**0.6628**	0.6588	0.6620
Glass	68.72	70.12	70.12	**73.32**	0.5645	0.5797	0.5809	**0.6250**
Led7digit	62.17	62.60	**65.42**	64.67	0.5775	0.5822	**0.6139**	0.6082
Lymphography	**83.19**	**83.19**	**83.19**	**83.19**	**0.6686**	**0.6686**	**0.6686**	**0.6686**
Nursery	**97.92**	**97.92**	97.84	**97.92**	**0.9695**	**0.9695**	0.9683	**0.9695**
Penbased	**98.19**	98.10	98.10	**98.19**	**0.9799**	0.9789	0.9789	**0.9799**
Satimage	86.79	86.95	**87.25**	**87.25**	0.8364	0.8383	**0.8422**	**0.8442**
Segment	97.32	97.36	97.27	**97.66**	0.9687	0.9692	0.9682	**0.9727**
Shuttle	97.43	98.03	98.76	**99.50**	0.9281	0.9441	0.9652	**0.9871**
Vehicle	**84.53**	84.40	84.41	84.17	**0.7936**	0.7920	0.7920	0.7889
Vowel	98.28	98.08	98.59	**98.79**	0.9811	0.9789	0.9844	**0.9867**
Yeast	60.25	59.98	60.92	**61.33**	0.4798	0.4753	0.4888	**0.4978**
Average	83.18	84.01	84.26	**84.42**	0.7666	0.7764	0.7807	**0.7844**

Table 2.2: Average 5-fold accuracy and kappa in test for methods PC, DCS, DRCW and KDEMRP† with PDFC as the base classifier over 16 KEEL datasets.

Method	All Methods	DCS,DRCW vs KDEMRP†	PC, DRCW vs KDEMRP†	PC, DCS vs KDEMRP†	PC, DCS vs DRCW
PC	45.34 (0.00040**)	-	35.31 (0.00004**)	33.56 (0.00007**)	32.12 (0.00459**)
DCS	38.09 (0.01305**)	32.03 (0.00387**)	-	26.81 (0.00569**)	24.34 (0.13958+)
DRCW	26.38 (0.34724)	24.78 (0.10201+)	23.94 (0.05033*)	-	17.03
KDEMRP	20.19	16.68	14.25	13.12	-

Table 2.3: Friedman aligned-rank tests comparing PC, DCS, DRCW and KDEMRP† with accuracy. A '+' near the p-value means that there are statistical differences with $\alpha = 0.15$ (85% confidence), a '*' with $\alpha = 0.10$ (90% confidence) and a '**' with $\alpha = 0.05$ (95% confidence)

Method	All Methods	DCS,DRCW vs KDEMRP†	PC, DRCW vs KDEMRP†	PC, DCS vs KDEMRP†	PC, DCS vs DRCW
PC	44.38 (0.00054**)	-	35.31 (0.00004**)	32.75 (0.00010**)	31.22 (0.00639**)
DCS	39.47 (0.00540**)	31.90 (0.00449**)	-	28.06 (0.00190**)	25.66 (0.06806*)
DRCW	26.44 (0.30742)	24.81 (0.10468+)	23.94 (0.05033*)	-	16.62
KDEMRP†	19.72	16.78	14.25	12.69	-

Table 2.4: Friedman aligned-rank tests comparing PC, DCS, DRCW and KDEMRP† with kappa. A '+' near the p-value means that there are statistical differences with $\alpha = 0.15$ (85% confidence), a '*' with $\alpha = 0.10$ (90% confidence) and a '**' with $\alpha = 0.05$ (95% confidence)

However, in order to see if the performance improvement is statistically significant, we have to refer to tables 2.3 and 2.4, which showcase the *Friedman aligned-rank* test results for accuracy and kappa respectively. In both these tables, the statistical comparison tests are carried out for all $\binom{4}{3}$ combinations of the methods as well as for all 4 methods at once. The key observations are as follows:

- KDEMRP† always gives a statistically significant inprovement with 95% confidence over *PC* and *DCS* with respect to both accuracy and kappa.
- *DRCW* also gives a statistically significant improvement over *PC* with 95% confidence for both metrics. However, it only gives an improvement over *DCS* with 90% confidence for kappa and 85% confidence for accuracy.
- For both metrics, KDEMRP† gives an improvement with 90% confidence over *DRCW* when *PC, DRCW* and KDEMRP† are separately compared.
- For both metrics, KDEMRP† gives an improvement with 85% confidence over *DRCW* when *PC, DCS* and KDEMRP† are separately compared.
- KDEMRP† however does not give a statistically significant improvement over *DRCW* when all 4 methods are compared.

Finally, table 2.5 shows the results of the *Wilcoxon signed-rank* tests for comparison between *DRCW* and KDEMRP†. From here, we can see that KDEMRP† gives a statistically significant improvement over *DRCW* with 90% confidence for both metrics accuracy and kappa.

Method	Accuracy	Kappa
DRCW	36.5	30.0
KDEMRP†	99.5 (0.09799*)	90.0 (0.08322*)

Table 2.5: Wilcoxon signed-rank tests comparing DRCW and KDEMRP† for both metrics accuracy and kappa. A '*' near the p-value means that there are statistical differences with $\alpha = 0.10$ (90% confidence)

2.5.5 Discussion

KDEMRP† gives a statistically significant improvement in terms of both accuracy and kappa over all 3 competitors *PC, DCS* and *DRCW*. The only high *p-value* encountered was for *DRCW* when all 4 methods were simultaneously compared. This highlights the instability of the *p-value* for *Friedman aligned-rank* tests expecially when multiple competing methods are added to the pool used for rankings. However, the *Wilcoxon signed-rank* test which directly compares *DRCW* and KDEMRP† establishes a statistically significant improvement for KDEMRP† over *DRCW* in terms of both accuracy and kappa.

It is worth mentioning that we also tried using *SVM* as the base classifier (both the SVM_{Poly} and SVM_{Puk} variants with parameters as mentioned in [33].) For some of these datasets (particularly the ones with nominal attributes), SVM_{Poly} performed

very poorly even as a binary classifier (In some cases, the *SVM* test performance between a pair of classes yielded less than 50% accuracy). As a result there were occasions where our approach actually worsened overall multi-classification performance, since we compute density estimates from training data *SVM* confidences. (Similar trends were observed with SVM_{Puk} as well and it typically performed worse than SVM_{Poly}).

PDFC (Positive Definite Fuzzy Classifier) as used in [33] interestingly uses SVM_{Poly} internally with the exact same parameters. After SVM_{Poly} is trained, a Gaussian *PDRF (Positive Definite Reference Function)* is used to supply fuzzy rules which transform the decision boundaries of each *SVM*. This transformation is key; although the final *PDFC* outputs are binary, the overall transformation of decision boundaries is highly non-linear with respect to the original *SVM* and significantly improves binary classification performance.

It can be argued that the performance of kernel *SVMs* could be significantly improved if the parameters used are optimized for each dataset by cross-validation over training data. However, this would require a full grid search over the kernel and loss parameters, and one would also have to do a nested cross-validation (since the test protocol also involves cross-validation). Since this would be very computationally expensive and our goal is not the comparison of binary classifiers, we skipped this.

It is important to note that *KDEMRP* is not tied to any specific binary classifier. *KDEMRP* can use a variety of underlying transformations: *SVM, PLS, CDF, DRCW* or any other, as long as they produce *soft decisions*. Moreover, *DRCW* is

fundamentally very different from *KDEMRP* when viewed as an *ensemble method*:

- The *DRCW* weight matrix (which is used to reweight confidences) is independent of the base classifier and relies entirely on the configuration of samples in the original feature space. In that sense, *DRCW* can potentially improve results for binary classifiers which even misclassify a majority of test samples.

- The *DRCW* weight matrix can be decomposed into a pair of weights for each *OVO* binary classifier. As a result, *DRCW* in conjunction with any other binary classifier can be seen as yet another binary classifier which outputs soft confidences. When viewed this way, the *ensemble method* for *multi-classification* is actually *WV (weighted voting)*, since this is how the final *multi-class* output is obtained from the reweighted DRCW confidence matrix.

- Thus, when we compare KDEMRP† against *DRCW* with *PDFC* as the base classifier, what we are actually comparing is *KDEMRP* against *WV* with *PDFC+DRCW* as the base classifier.

Finally, we conclude this section by saying that *"KDEMRP when combined with DRCW is able to improve the usage of DRCW alone."*

2.6 Experiments on Computer Vision Datasets

We also test *KDEMRP* on 6 real-world computer vision datasets. We compare our approach against the *VOTE* and *NEST*, which are the most commonly used off-the shelf *OVO* ensemble methods. We use 2 base classifiers: *Linear SVM* and

CDF. We use classification accuracy as our performance metric, and similar to the previous section, we do the *Friedman aligned-rank* test followed by the *Holm* post-hoc method separately for *Linear SVM* and *CDF* in order to judge the statistical significance of our approach.

We did not pursue *DCS* or *DRCW* for these experiments since we found it computationally intractable. Few of these datasets have very high dimensional feature spaces (prohibitively expensive nearest neighbor searches) coupled with very lengthy test protocols such as LOGO (Leave One Group Out).

2.6.1 Base Classifiers

We use 2 binary classifiers : *Linear SVM* and *CDF* for our *OVO ensemble* experiments on the computer vision datasets. The *Composite Discriminant Factor Analysis (CDF)* classifier, which we introduced in [46], was shown to outperform *Linear SVM* in several challenging detection/ classification tasks. We briefly describe *CDF* below :

CDF is originally a dimensionality reduction technique, based on *Partial Least Squares (PLS)*. *PLS* based methods work in an iterative fashion, by finding the direction of maximum covariance between the data and labels, deflating the data with respect to that direction and then recomputing the next direction of maximum covariance. This process is repeated until a desired number of *PLS* directions have been obtained. The projection of the original data on to these *PLS* directions, serves as a dimensionality reduction process, which retains components of the data that

have maximum covariance with respect to the labels. The number of *PLS* factors to be considered is usually determined by cross-validation if the subsequent task is classification.

In *CDF*, a linear combination of *PLS* directions is computed and is referred to as a *composite*. The original data is then deflated by the *composite*, *PLS* is applied on the deflated data to find the second composite, and so on, until a desired number of composites have been found. So, *CDF* can be viewed as a nested variant of *PLS*. The *composites* can then be used the same way one would use the *PLS* directions. The advantage this method offers over *PLS* is that fewer *composites* are required to reliably represent the data is much less, leading to a much better dimensionality reduction for the original data.

CDF can also be used as a binary classifier directly, with the projection direction being the linear combination of the *composites* (or simply the first *composite*). Results in [46] show that a classifier based on the first *composite* itself provides excellent detection and classification performance.

The default parameters of $C = 1$ are used for *Linear SVM*. For *CDF*, there are no parameters to be specified. The number of *PLS* factors to be used in *CDF* is automatically determined from the training data.

2.6.2 MIT67 Scene Recognition dataset

The *MIT67* [47] dataset consists of 15620 images which span 67 categories of indoor scenes. These scenes comprise of various types of shops, residences, offices,

leisure places and public spaces. The *Convolutional Neural Network (CNN)* model *OverFeat* [48] which is pre-trained on the *ImageNet* [49] *ILSVRC* 2013 dataset, is used to compute 4096 dimensional *CNN* features for each image.

We used precomputed *CNN* features from [50] and the same test/train split for evaluation as [47]. A subset of 5360 images (80 from each class) are used for training and 1340 images (20 from each class) are used from testing.

2.6.3 UCF50 Action Recognition Dataset

The *UCF50* [51] dataset consists of realistic videos taken from *youtube*, which span 50 different action categories. For all categories, each video is represented by a 14965 dimensional vector using action-bank [52] features. Each category contains around 120-160 samples. The samples within each class are further sub-classified into groups. For any class, each group corresponds to a unique *youtube* video, and all the samples within that group are essentially small 2-5 second video clips extracted from different temporal regions in the video. There are 25-30 groups per class, and around 4-8 samples within each group.

We use *Leave One Group Out (LOGO)* cross-validation, as done in [53], to report results. *LOGO* is similar to *Leave One Out (LOO)*, with the only difference being that a *group* of samples that are not independent (*i.e.*clips from the same video) is left out for testing in every *LOGO* iteration, as opposed to just leaving a single sample out in *LOO*. We perform 2 sets of experiments, one with all 50 classes, and the other with only those 10 classes which were the most confused with respect

to the reported baseline for this dataset [52].

2.6.3.1 Object Recognition Datasets

We also use 3 computer vision datasets (*Birds, Butterflies, Robotics*) to test our proposed *KDEMRP OVO ensemble* approach.

Birds [54], **Butterflies** [55] are instance recognition datasets. The *Birds* dataset contains 600 images of 6 different birds, while the *Butterflies* dataset contains 618 images of 7 different butterflies. Both datasets have the same feature representation (1000 *HSV* + 1000 dense *SIFT* + 680 Pyramidal *HOG* features = 2680 features per image).

The **Robotics** [56] dataset contains 31 object classes with only 783 samples in total. Each sample is represented by 92 features (37 *dslr* + 55 *webcam* features).

Results for *Birds, Butterflies* and *Robotics* are reported via 5 fold cross-validation, based on the splits provided along with the respective datasets.

2.6.4 Results

The results for all 6 computer vision datasets is summarized in table 2.6. Linear SVM outperforms CDF for *Birds, Butterflies*, while *CDF* outperforms *Linear SVM* for *UCF50* dataset. For the *Robotics* dataset, *CDF* outperforms Linear SVM with *VOTE/NEST ensembles*, but Linear SVM outperforms *CDF* with *KDEMRP* as the ensemble method. For the *MIT67* dataset, although Linear SVM outperforms *CDF* with *VOTE/NEST ensembles*, both classifiers give significantly improved yet

Data-sets	Linear SVM			CDF		
	VOTE	NEST	KDEMRP	VOTE	NEST	KDEMRP
MIT67	58.13	58.27	**65.52**	54.92	55.37	**65.60**
UCF50 (10 hardest classes)	66.89	66.92	**68.95**	66.99	67.22	**69.14**
UCF50 (50 classes)	61.48	61.61	**63.55**	62.70	62.84	**64.64**
Birds	55.22	55.78	**57.56**	54.32	**54.54**	52.56
Butterflies	61.58	61.89	**62.08**	59.47	59.36	**61.07**
Robotics	32.58	32.63	**44.39**	40.06	40.19	**42.07**

Table 2.6: Comparison of average classification accuracies for *MIT67, UCF50, Birds, Butterflies and Robotics* datasets, for the proposed *KDEMRP* method with most commonly used *OVO ensemble* methods *VOTE, NEST*; for 2 binary classifiers *Linear SVM* and *CDF*. The best performing ensemble method for each binary classifier is highlighted.

roughly similar results with *KDEMRP* as the ensemble method. The important thing to note from these trends, is that for both *Linear SVM* and *CDF*, the proposed *KDEMRP ensemble method* outperforms the state-of-the-art *ensemble* methods *VOTE, NEST* for all datasets, except for the *Birds* dataset with *CDF* as the binary classifier.

Method	Linear SVM	CDF
VOTE	15.57 (0.00097*)	14.29 (0.02496+)
NEST	13.43 (0.00447*)	12.71 (0.04293+)
KDEMRP	4	6

Table 2.7: Friedman aligned-rank tests comparing VOTE, NEST and KDEMRP with accuracy. A '+' near the p-value means that there are statistical differences with $\alpha = 0.05$ (95% confidence) and a '*' with $\alpha = 0.005$ (99.5% confidence)

Table 2.7 shows the results of the *Friedman aligned-rank* statistics test applied separately for *Linear SVM* and *CDF*. The *p-values* show that *KDEMRP* gives statistically significant improvements over *VOTE* and *NEST* with a confidence of 95% for *CDF* and 99.5% for *Linear-SVM*.

2.6.5 Discussion

Linear SVM, which is a *max-margin* classifier operates very differently from *CDF*, which is a *partial least squares (PLS)* based classifier. A *Linear SVM* binary classifier bases its decision boundary exclusively on a selected number of samples known as *support vectors*. On the other hand, *CDF* considers all the training sam-

ples, and determines its decision boundary based on a linear combination of *PLS* directions, which are in turn based on *maximizing covariance* of the data with the labels. The superior performance of our proposed method *KDEMRP* over *VOTE* and *NEST* for both *Linear SVM* and *CDF*, indicates a good generalization ability of our method over the choice of binary classifier. This is very important, since no one binary classifier performs the best for all datasets, in practice. *PLS* based classifiers, such as *CDF*, tend to outperform *Linear SVM*, when lots of quality training data is available with minimal outliers [46]. However, when the data is either sparse, or is known to have lots of potential outliers, *max-margin based* classifiers such as *Linear SVM* are usually a safer bet.

One of the reasons for the good generalization of *KDEMRP* over different classifiers is due to the underlying *KDE* process. The success of the method boils down to getting good probability estimates from the ensemble of classifier scores and *KDE* automatically adjusts itself to deal with the different scales, distortions and range of scores for different classifiers.

It is also worth noting that although linear binary classifiers are used in the *ensembles*, the final multi-classification decision boundary induced by the *KDEMRP* method could be highly non-linear. This is due to *KDE* inherently being capable of handling complex non-linear density estimates, which makes *KDEMRP* an extremely high capacity classifier, capable of handling almost any shatter of the training data. As a result, *KDEMRP* can even be used to solve multi-classification problems, which are *not* linearly separable, despite using linear classifiers for the *ensembles*.

2.7 Summary

We present a robust probabilistic *OVO ensemble* method *KDEMRP* that can be used with any generic binary classifier to solve a multi-classification problem. The novelty of this approach lies in the non-parametric estimation of multi-class probabilities directly from all the pairwise binary classifier scores as a joint density. The feasibility and success of the method is largely attributed to the univariate approximation to multi-variate *KDE* based on *PCA projections*. Multi-variate *KDE* requires not only more computations, but also a significantly higher number of training samples to obtain reliable estimates, which most often makes its direct application infeasible for data sets with small number of samples. Experimental results on the KEEL datasets show that *KDEMRP* when combined with the *state-of-the-art DRCW* method significantly improves results. Experimental results on the Computer Vision datasets show that *KDEMRP* significantly statistically outperforms the most commonly used *ensemble* methods *VOTE, NEST* for 2 very different binary classifiers Linear SVM and *CDF*.

Chapter 3: Generalized Deep Image to Image Regression

3.1 Overview

Over the last few years, generic deep convolutional neural network (DCNN) architectures such as variants of VGG [8] and ResNet [1] have been immensely successful in tackling a diverse range of classification problems and achieve state-of-the-art performance on most benchmarks when used out of the box. The key feature of these architectures is an extremely high model capacity along with a robustness to minor unwanted (*e.g.*translational/rotational/illumination) variations. Given suitable training data, such models can be discriminatively trained in a reliable end-to-end fashion. However, since classification tasks only require a single (potentially multi-variate) class label corresponding to the entire image, early architectures focused solely on developing strong global image features.

Semantic Segmentation was one of the first applications to witness the extension of DCNNs to output dense pixel wise predictions [57–61]. These approaches used either VGG or ResNet (without the fully connected layers) as their backbone and introduced architectural changes such as skip layers [57], deconvolutional networks [58, 62], hypercolumns [61] or laplacian pyramids [60] to facilitate the retention/reconstruction of local input-output correspondences. While these approaches

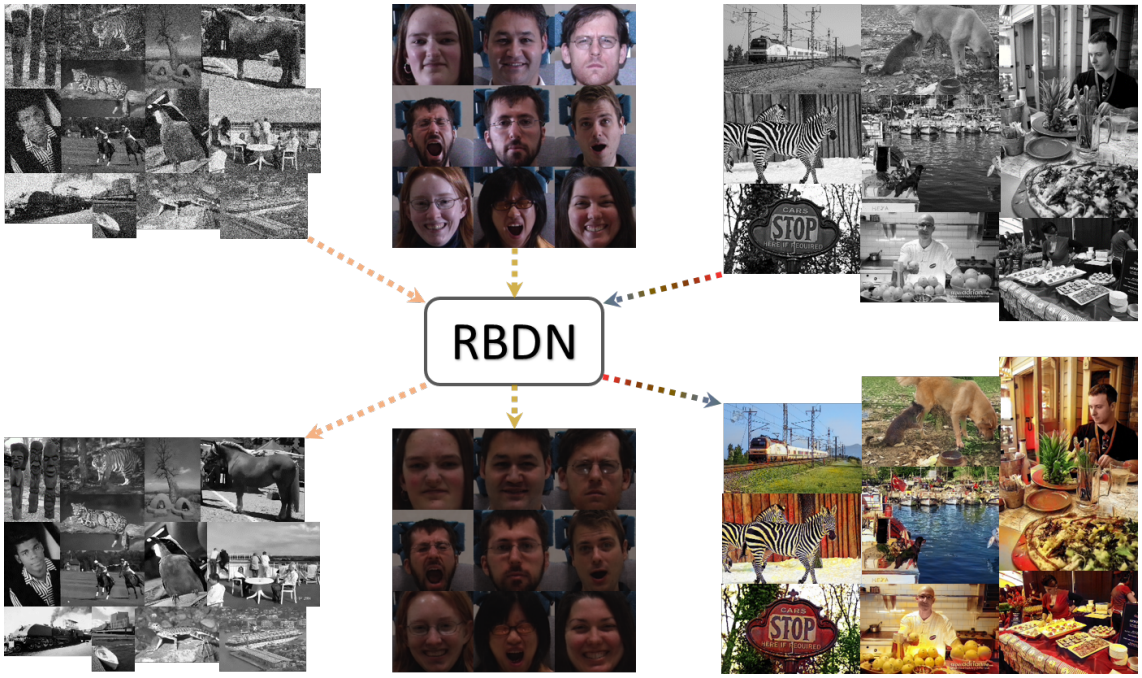

Figure 3.1: Proposed **RBDN** used for diverse Im2Im regression tasks: (from left to right) **Denoising, Relighting, Colorization**.

performed very well on segmentation benchmarks, they introduced a trade-off between locality and context. Since the task still remained one of classification (albeit at a pixel level), the trade-off was skewed in favor of incorporating more context and subsequently reconstructing local correspondences from global activations. This is perhaps why some of these approaches had to rely on ancillary methods such as Conditional Random Fields (CRFs) [58, 59] to enhance the granularity of their predictions.

Image-to-Image (Im2Im) regression entails the generation of dense "continuous" pixelwise predictions, where the locality-context trade-off is highly task-dependent (typically skewed more in favor of locality). Several DCNN based approaches have been proposed for specific Im2Im regression tasks such as denoising, relighting, colorization, *etc.*. These approaches typically involve highly task-specific architectures coupled with fine-tuned ancillary post processing methods. However, unlike classification DCNNs, no truly generic architecture for Im2Im regression has yet been proposed which performs consistently well on a diverse range of tasks. It is perhaps the task-dependent locality-context trade-off coupled with the habitual trend of incorporating VGG/ResNet architectures for non-classification tasks, that have impeded progress in this regard.

We propose a generic Im2Im DCNN architecture, **RBDN**, which eliminates this trade-off and automatically learns how much locality/context is needed based on the task at hand, through the *early development* of a cheaply computed rich multi-scale image representation using recursive multi-scale branches, learnable upsampling and extensive parameter sharing.

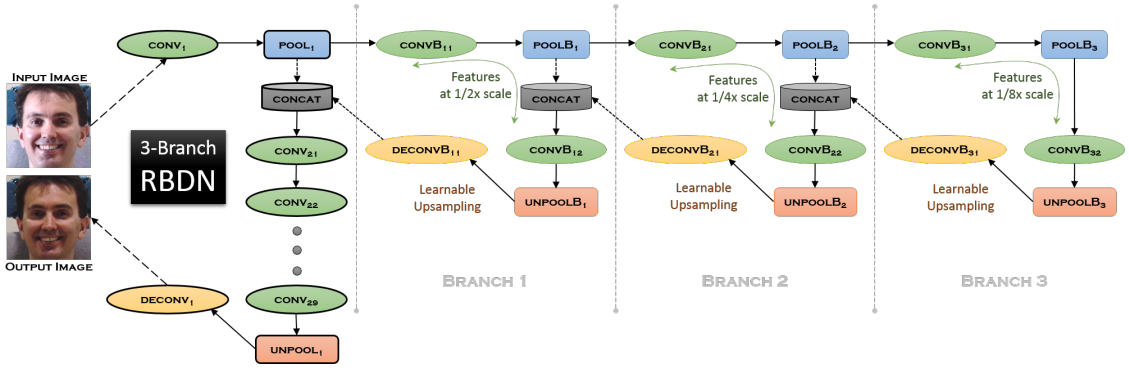

Figure 3.2: Architecture of proposed generic RBDN approach with 3 branches. The various branches extract features at multiple scales. Learnable upsampling with efficient parameter sharing is used to recursively upsample the activations for each branch until it merges with the POOL1 output, leading to a cheap multi-context representation of the input. This multi-context map is subjected to series of 9 convolutions which can supply ample non-linearity and automatically choose how much context is needed based on the task at hand.

3.2 Related Work

We first describe two recently proposed Im2Im DCNN approaches [3,63] which also have a fairly generic architecture and compare the similarities and differences with our proposed RBDN approach. We then describe some of the related work specific to relighting, denoising and colorization.

3.2.1 Generic Im2Im Regression

Deep End-2-End Voxel-2-Voxel prediction [63] proposed a video-to-video regressor for solving 3 tasks: semantic segmentation, optical flow and colorization. Their architecture consists of a VGG [8] style network on which they add branches which upsample and merge activations. Unlike Hypercolumns [61], they make the upsampling learnable and perform it in a more efficient way with weight sharing. While [63] use upsampling to recover local correspondences, DnCNN [3] on the other hand entirely eliminate downsampling and use a simple 18 layer fully convolutional network with residual connections for handling 3 tasks: denoising, super-resolution and jpeg-deblocking. Our proposed RBDN architecture can be viewed as a hybrid of [3, 63]. While we do utilize multi-scale activations like [63], we do so very early in the network and generate a cheap composite multi-context representation for the image. Subsequently, we pass the composite map to a linear convolution network like [3].

3.2.2 Face Relighting

In the field of Face Recognition/Verification, while most research focuses on extracting illumination-invariant features, *relighting* is the relatively less explored alternative [64] of directly making illumination corrections/normalizations to an image. Traditional face relighting approaches used the Retinex [65]/Lambertian Reflectance [66] theory and used spherical [66, 67]/hemispherical [68] harmonics, subspace-based [69, 70] or dictionary-based [71–76] illumination corrections. Deep Lambertian Networks [77] encoded lambertian models/illumination corrections directly into their network architecture. This however limited the expressive power of the network, particularly due to the strong lambertian assumptions on isotropicity and absence of specular highlights, which seldom hold true for face images. In section 3.4.2, we show that it is possible to train a well-performing relighting model without making any lambertian assumptions using our generic RBDN architecture.

3.2.3 Denoising

Denoising approaches typically assume an Additive White Gaussian Noise(AWGN) of known/unknown variance. Traditional denoising approaches include ClusteringSR [78], EPLL [79], BM3D [80], NL-Bayes [81], NCSR [82], WNNM [83]. Among these, BM3D [80] is the most popular, very well engineered and still widely used as the state-of-the-art denoising approach. Early DCNN based denoising approaches [84–88] required a different model to be trained for each noise variance, which limited their practical use. Recently, a Gaussian-CRF based DCNN approach (DCGRF [89])

was proposed which could explicitly model the noise variance. DCGRF could however only reliably model noise levels within a reasonable range and had to use two models: low-noise DCGRF ($\sigma < 25$) and high-noise DCGRF ($25 \leq \sigma \leq 50$). In section 3.4.3, we show that a single model of our proposed RBDN approach trained on a wide range of noise levels ($\sigma \leq 50$) achieves competitive results and outperforms all the previously proposed approaches at all noise levels $\sigma \in [25, 55]$.

3.2.4 Colorization

The inherent color ambiguity in a majority of objects makes colorization a very hard and ill-posed problem. Early works on colorization [90–97] required a reference color image from which the color of local patches in the input image was inferred through parametric/non-parametric approaches. Only recently, have DCNN approaches [98–101] been used to solve colorization as an Im2Im classification/regression problem from grayscale to color without requiring auxiliary inputs. [98, 100] use Hypercolumns [61], while [99] use a complex dual-stream architecture that simultaneously identifies/classifies object classes within the image and uses class labels to colorize the input greyscale image. The classification branch of their network is identical to VGG [8], while the colorization branch of their network mimics the DeconvNet [58] architecture. The best colorization results however are obtained by [101] despite using a fairly simple VGG [8] style architecture with dilated convolutions. The key contribution of [101] is their novel classification-based loss function over the quantized probability distribution of ab values in the Lab color

space. They further add a class re-balancing scheme that pushes the predictions away from the statistically likely gray colors, resulting in very colorful colorizations. In section 3.4.4, we use the same loss function as [101] but replace their VGG-style architecture with our proposed RBDN architecture and obtain excellent colorizations.

3.3 Generic Im2Im DCNNs

Many Im2Im approaches use VGG/ResNet as their backbone because of their effectiveness and availability. However, this leads to suboptimal architectures (3.3.1) for these types of tasks because of the inherent bias towards including more context at the expense of sacrificing locality. We instead propose RBDN (3.3.2) which uses recursive branches to obtain a cheap multi-context locality-preserving image representation very early on in the network. In sections 3.3.3, 3.3.4, 3.4.2.1, we describe our network architecture in more detail and analyze its various components.

3.3.1 Classification DCNNs are a bad starting point

Classification DCNNs typically contain a multitude of interleaved downsampling layers (max-pooling or strided convolutions) which ultimately squash the image to a 1-D vector. With GPU memory being the major bottleneck for training DCNNs, downsampling layers enable the exploration of very deep architectures while providing a natural translational invariance. However, problems arise when attempting to directly port these networks for Im2Im regression tasks. Design changes are

needed for retention/recovery of local correspondences, as these get muddled across channels in the middle layers. Recovery with repeated upsampling is inevitably a lossy process, which is particularly harmful for regression tasks demanding continuous pixelwise predictions. Alternatively, local correspondences can be retained (*e.g.*skip layers, hypercolumns) by merging activation maps from earlier layers at the penultimate layer. The downside to this approach is that activations from very early layers (which contain the bulk of the local correspondences) have a poor capability to model non-linearity, which limits the overall capacity of the network for modeling localized non-linear transformations. For a DCNN to be successful as a generic Im2Im regressor, it would necessarily need to maintain local pixelwise features, each of which develop strong global representations across the pipeline while independently preserving local information.

3.3.2 Proposed Approach: RBDN

Figure 3.2 shows the architecture for our proposed Recursively Branched Deconvolutional Network with three branches. At a high level, the network first extracts features at scales 1(max-locality), $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$(max-context) and merges all these activations early on to yield a composite map, which is then subjected to a series of convolutions (non-linear transformation) followed by a deconvolution (reconstruction) to yield the output image. The key feature of this network is the multi-scale composite map and how it is efficiently generated using recursive branching and learnable upsampling. During training, the network has a broad locality-context

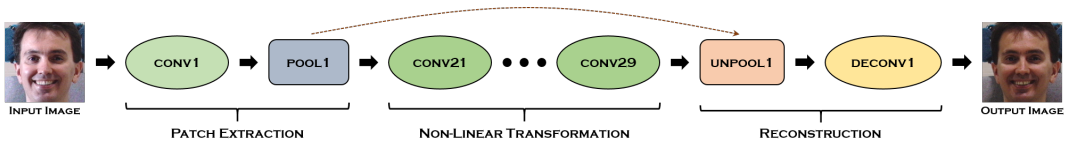

Figure 3.3: Architecture of linear 9-64-3-9 net B_0.

spectrum to work with early on. The series of convolution layers that follow suit can choose the amount of context based on the task at hand and apply ample non-linearity. This translates to a range of modeling capabilities: anywhere from context-aware regression maps to highly localized non-linear transformations (which were difficult to achieve with previously proposed DCNNs).

Our generic K-branch RBDN network has two major components: the main branch B_0 (which serves as the backbone of the network) and the recursive branches $(B_1, ..., B_K)$ (which serve as the head of the network).

3.3.3 The Linear Base Network B_0

Inspired by traditional sparse coding approaches, we approach the Im2Im regression problem with a simple network (denoted by its parameters K-c-T-D) having three distinct phases:

- *Patch Extraction*: conv (K x K x c) + max-pooling
- *Non-Linear Transform*: D conv layers (T x T x c)
- *Reconstruction*: unpooling(using max-pool locations) + deconvolution ($K \times K \times c$)

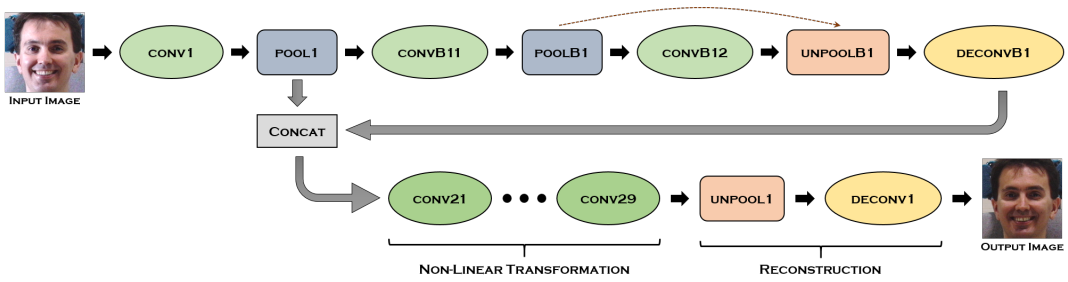

Figure 3.4: Adding the first branch to B_0.

We use ReLU [102] as the activation function and use a batch normalization [103] layer after each convolution/deconvolution. We independently experimented with values K, c, T, D while performing our relighting experiments and found that increasing K, c, T only yields a minor improvement, while increasing the network depth D yielded a significant monotonic improvement until 9 convolution layers, after which performance saturated. Our final network that yielded the best results is shown in figure 3.3. We denote this network as B_0 from here on. (We will use it as the main branch for all RBDN networks).

3.3.4 Recursive Branches $B_0, ...B_K$

While the base network B_0 by itself performs well for relighting, one of its limitations is a very low field of view. Unlike conventional DCNNs, we cannot add downsampling midway since this would corrupt our local correspondences. As a result, we keep B_0 and its local correspondences intact and instead add a branch B_1 to the network (see figure 3.4) at the first pooling layer.

Within B_1, CONVB_{11}+POOLB_1+CONVB_{12} computes features at half the

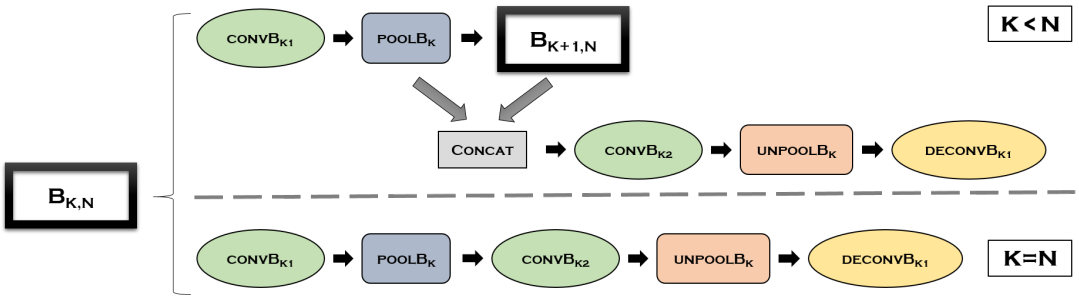

Figure 3.5: Defining the recursive branch module $B_{K,N}$. In the top half, the box with the thick black border, $B_{K+1,N}$ contains the recursive branch. The bottom half of the figure shows the base case (the last branch that does not contain any recursion).

scale and UNPOOLB$_1$+DECONVB$_{11}$ provides a learnable upsampling. The output of B_1 is then merged with B_0 at POOL1 itself, which gives the remainder of the network (which invoke the bulk of non-linearity) access to features at 2 different scales.

We can generalize B_1 to multiple branches $B_1, ...B_K$. In order to do so, we start by defining the recursive branch module $B_{K,N}$ in figure 3.5 which corresponds to the K^{th} branch in a N-branch network. Note that branch $B_{K+1,N}$ originates and merges within branch $B_{K,N}$. The advantage of such a recursive construction is two-fold:

- Activations from deeper branches would have to be upsampled many times before merging with the main branch. The recursive construction helps deeper branches partially benefit from the learnable upsampling machinery in the shallow branches.

- Aside from the benefit of parameter sharing, the recursive construction forces activations from deeper branches to traverse a longer path, thus accruing many ReLU activations. This enables deeper branches to model more non-linearity, which is beneficial since they cover a larger *field of view* and correspond to global features.

3.4 Experiments

We train our generic RBDN architecture for three diverse tasks: relighting, denoising and colorization. We train all our models on a Nvidia Titan-X GPU and use the Caffe [104] deep learning framework. For our denoising/colorization experiments, we augment Caffe with utility layers for noise policies (adding WGN to input with σ randomly chosen within a user specified range) and image conversions (RGB to YCbCr/Lab space), which streamline the training procedure and enable the use of practically any image dataset out of the box without any pre-processing. We use ReLU [102] as the activation function and perform Batch Normalization [103] after every convolution/deconvolution layer in all RBDN models. All our RBDN models and code are publicly available at `https://github.com/venkai/RBDN`.

Unless otherwise mentioned, we train our RBDN models with the mean square error (MSE) as the loss function, crop size of 128 (chosen randomly from the full-sized training images without any resizing), learning rate of 1e-7, mini-batch size of 64, step-size of 100000 and train our model for 500000 iterations using Stochastic Gradient Descent [105] (SGD) with momentum and weight decay. During inference,

the network by virtue of being fully convolutional can handle variable sized inputs. Inference takes ~1s on a 320x480 image. Training takes ~1 day for relighting and ~2 weeks each for colorization/denoising.

3.4.1 Training Datasets

CMU-MultiPIE [4]: Face images of 337 subjects are recorded over 4 sessions. Within a session, there are face images of each subject exhibiting 13 pose x 19 illumination x 2-3 expression variations. We used images of 208 subjects which did not appear in all sessions for training our relighting RBDN, and images of 64 other subjects for validation.

ImageNet ILSVRC2012 [49]: 1.2 million training images and $150,000$ images each for validation and test.

MS-COCO [11]: $80,000$ training images and $40,000$ images each for validation and test.

For training both our denoising/colorization RBDN, we fuse the train/validation sets of both ImageNet and MS-COCO (total of 1.47 million training images).

3.4.2 Face Relighting

We train our relighting RBDN on 20786 images from CMU-MultiPIE, which takes as input a frontal face image with varying illumination and outputs the image with only ambient lighting. We used a crop size of 224, step-size of 12000 and trained our model for 40000 iterations.

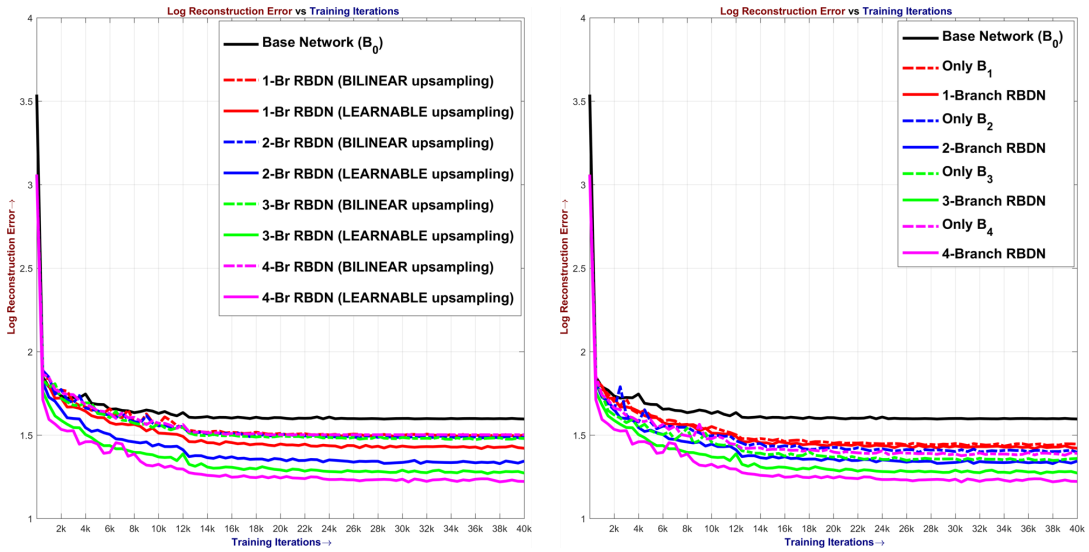

Figure 3.6: Analysing the effect of learnable upsampling(left) and recursive branching(right). Error plots on the CMU-MultiPIE [4] validation set show a positive influence for both learnable upsampling and recursive branching.

3.4.2.1 Analysis of RBDN

Compared to the base network B_0, a K-branch RBDN has two major additions: the recursive branching and learnable upsampling. We perform two sets of relighting experiments to independently observe the efficacy of both on a K-branch RBDN($K = 0, 1, 2, 3, 4$) as follows:

- We removed the CONCAT layers which merge the different branches. This resulted in a linear network (B_K only) similar in structure to the deconvolutional networks used for semantic segmentation [58, 62].
- We replaced the learnable upsampling with fixed bilinear upsampling.

Figure 3.6 shows the error plots of log reconstruction error on the CMU-MultiPIE [4]

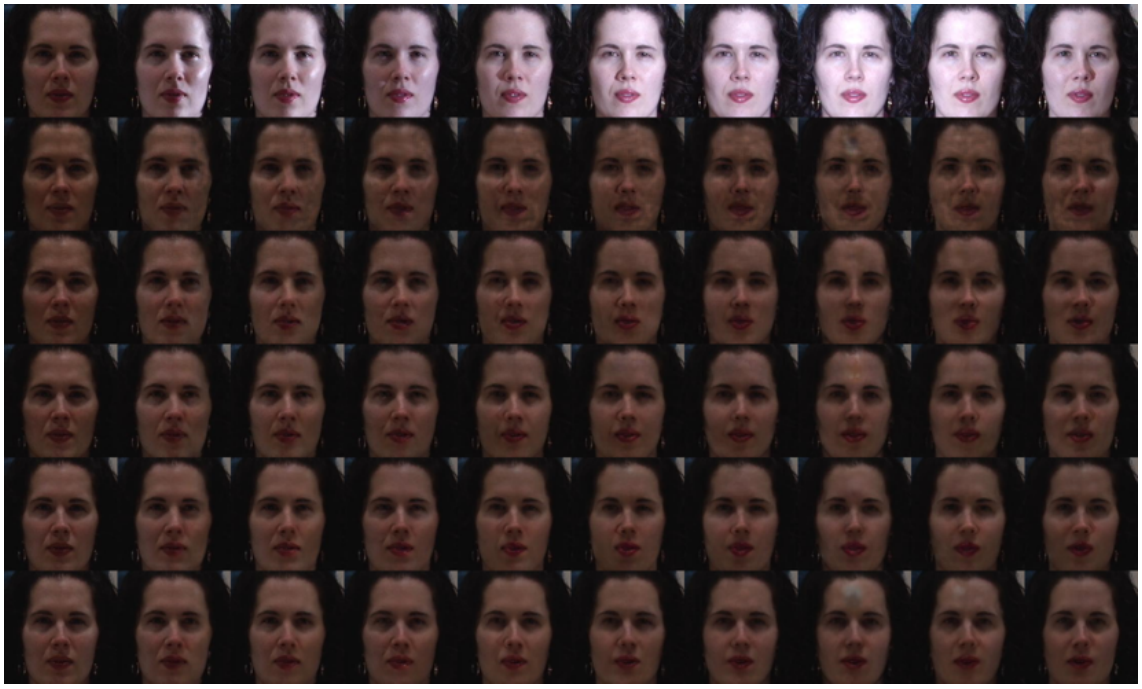

Figure 3.7: Relighting RBDN results for a subject from the CMU-MultiPIE [4] validation set. **Top Row:** Input images (ground truth is top-left image). 2^{nd} **row:** B_0 output (no branches; strong artifacts can be seen.) 3^{rd}-6^{th} **row:** RBDN outputs for $1, 2, 3, 4$ branches respectively. Results improve with increase in number of branches up to 3 branches. The network starts overfitting at 4 branches.

validation set vs training iterations for both experiments. The plots show that both learnable upsampling and recursive branching independently have a positive impact on performance.

3.4.3 Denoising

We train a single 3-branch RBDN model for denoising which takes as input a grayscale image corrupted by additive WGN with standard deviation uniformly

Figure 3.8: **Relighting results** on test sets. The goal is to render faces from various unknown lighting conditions to a fixed lighting condition. **Odd rows:** Inputs, **Even Rows**: 3-branch RBDN output. Note that the model is trained exclusively on *frontal* face images with *constrained* illumination variations from **CMU-MultiPie** [4], but still generalizes reasonably well to *unconstrained* face images in **Janus-CS0** [5] under a variety of *poses, illuminations, expressions, occlusions, affordances* (hats, glasses, *etc.*)

.

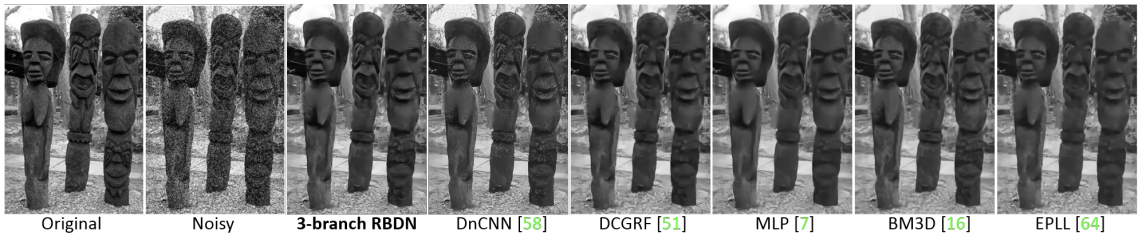

Figure 3.9: Visual comparison of various denoising approaches on a test image from BSD300 with WGN of $\sigma = 50$.

Test σ	10	15	20	25	30	35	40	45	50	55	60
ClusteringSR [78]	33.27	30.97	29.41	28.22	27.25	26.30	25.56	24.89	24.28	23.72	23.21
EPLL [79]	33.32	31.06	29.52	28.34	27.36	26.52	25.76	25.08	24.44	23.84	23.27
BM3D [80]	33.38	31.09	29.53	28.36	27.42	26.64	25.92	25.19	24.63	24.11	23.62
NL-Bayes [81]	33.46	31.11	29.63	28.41	27.42	26.57	25.76	25.05	24.39	23.77	23.18
NCSR [82]	33.45	31.20	29.56	28.39	27.45	26.32	25.59	24.94	24.35	23.85	23.38
WNNM [83]	**33.57**	31.28	29.70	28.50	27.51	26.67	25.92	25.22	24.60	24.01	23.45
TRD [106]	-	31.28	-	28.56	-	-	-	-	-	-	-
MLP [86]	33.43	-	-	28.68	-	27.13	-	-	25.33	-	-
DCGRF [89]	33.56	**31.35**	**29.84**	28.67	27.80	27.08	26.44	25.88	25.38	24.90	**24.45**
DnCNN [3]	33.32	31.29	**29.84**	28.68	27.70	26.84	26.05	25.34	24.68	24.05	23.39
3-branch RBDN	32.85	31.05	29.76	**28.77**	**27.97**	**27.31**	**26.73**	**26.24**	**25.80**	**25.22**	23.25

Table 3.1: Mean PSNR for various denoising approaches on 300 test images. A *single* denoising model is used to report all results for **RBDN** (trained on $\sigma \in [8, 50]$) and DnCNN [3] (trained on $\sigma \in [0, 55]$). For other comparison approaches, note that the best performing model at each noise level is used to report results.

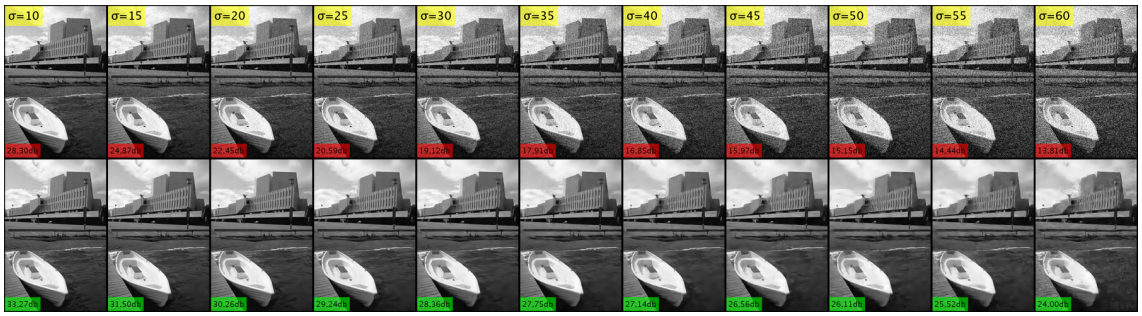

Figure 3.10: Illustrating the capability of a single RBDN model to handle a range of noise levels(yellow box). **Top Row:** Noisy test image (PSNR in red box). **Bottom Row:** Denoised result with 3−branch RBDN (PSNR in green box)

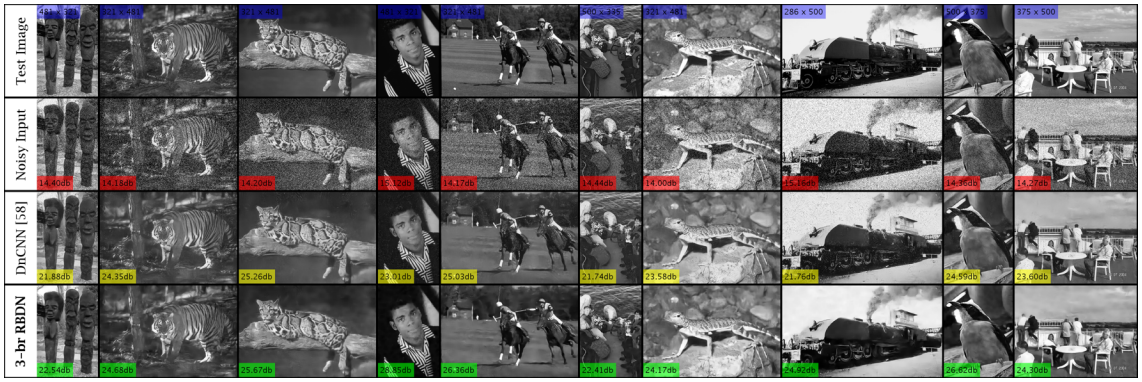

Figure 3.11: Illustrating RBDN's ability to reliably denoise at $\sigma = 55$, outside our training bounds ($\sigma \in [8, 50]$). The 18-layer DnCNN [3] (despite using $\sigma = 55$ for training) is outperformed by our 9-layer RBDN. Red, Yellow, Green boxes show the PSNR.

randomly chosen in the range $\sigma \in [8, 50]$. We use the same evaluation protocol as [89], with a 300 image test set (all 100 images of the BSD300 [107] test set and 200 images from PASCAL VOC2012 [10] dataset). Precomputed noisy test images from [89] are used to compare all approaches for a fair realistic evaluation.

3.4.4 Colorization

We first transform a color image into YCbCr color space and predict the chroma (Cb,Cr) channels from the luminance (Y-channel) input using RBDN. The input Y-channel is then combined with the predicted Cb,Cr channels and converted back to RGB to yield the predicted color image. We denote this model as **RBDN-YCbCr**.

Inspired by the recently proposed Colorful Colorizations [108] approach, we

Figure 3.12: Colorization results for images from MS-COCO test set. (Please see supplementary for more comparisons)

train another RBDN model which takes as input the L-channel of a color image in *Lab* space and tries to predict a 313-dimensional vector of probabilities for each pixel (corresponding to 313 *ab* pairs resulting from quantizing the *ab*-space with a grid-size of 10). Subsequently, the problem is treated as multinomial classification and we use a softmax-cross-entropy loss with class re-balancing as in [108]. Instead of SGD, we use the Adam [109] solver for training, with a learning rate of 3.16e-3 ($\gamma = 0.316$), step-size of 45000, mini-batch size of 128 and train our model for 200000 iterations. During inference, we use the annealed-mean of the softmax distribution to obtain the predicted *ab*-channels as in [108]. We denote this model as **RBDN-Lab**.

3.5 Results

Relighting: Figure 3.7 shows the RBDN outputs with $0, 1, 2, 3, 4$ branches for a subject from the CMU-MultiPIE validation set. The improvement in results

from B_0 (no branches) to 1-branch RBDN is very prominent, after which there is a gradual improvement with increase in number of branches up to 3. Results deteriorate when transitioning to a 4-branch RBDN (possibly due to overfitting on the relatively small dataset). We qualitatively evaluate our results (figure 3.8) on the test-sets of CMU-MultiPIE and Janus-CS0 [5] for the 3-branch RBDN. While RBDN achieves near perfect relighting on CMU-MultiPIE, it surprisingly generalizes well (without any finetuning) to unconstrained images in Janus-CS0.

Denoising: Table 3.1 shows the mean PSNR for various denoising approaches on the 300 benchmark test images. Besides RBDN, DnCNN [3] and DCGRF [89], all other approaches train a separate model for each noise level. For DCGRF [89], results are reported with a low noise model for test $\sigma \leq 25$ and a high noise model for test $\sigma \geq 30$. The results for both DnCNN [3] and our 3-branch RBDN however correspond to a *single* model trained to automatically handle *all* noise levels. Our model outperforms all the other approaches at test noise $\sigma \in [25, 55]$. Figure 3.9 shows a visual comparison of various denoising approaches for a test image from BSD300. Figure 3.10 highlights a single RBDN model's denoising capability across a range of noise levels. Figure 3.11 illustrates the generalization ability of the RBDN to reliably denoise at a very high noise level of $\sigma = 55$ (which is outside the bounds of our training). The fact that our 9-layer RBDN (without any residual connections [1]) outperforms the 18-layer residual DnCNN [3], suggests that cheap early recursive branching is more beneficial than added depth.

Colorization: Figure 3.12 shows the colorizations of various models on the MS-COCO test set. The $3, 4$-branch RBDN-YCbCr models produce decent coloriza-

tions, but are very dull and highly under-saturated. This is however not an architectural limitation, but rather the MSE loss function which tends to push results towards the average. Colorization is inherently ambiguous for a large majority of objects such as cars, people, animals, doors, utensils, *etc.*, several of which can take on a wide range of permissible colors. On the other hand, the MSE based models are able to reasonably color grass, sky, water as these typically take on a fixed range of colors. Softmax cross-entropy loss based models with class rebalancing ([108] and the 4-branch RBDN-Lab) are able to overcome the under-saturation problem by posing the problem as a classification task and forcibly pushing results away from the average. Finally, the only difference between the 4-branch RBDN-Lab and the linear dilated convolutional network of [108] is the architecture. Both models give very good colorizations, with one appearing better than the other for certain images and vice-versa, although the colorizations of RBDN-Lab have a higher saturation and appear slightly more colorful for all images.

3.6 Summary and Future Work

We presented a DCNN architecture for Im2Im regression: RBDN, which gives competitive results on 3 diverse tasks: relighting, denoising and colorization, when used off-the-shelf without any task-specific architectural modifications. The key feature of RBDN is the development of a cheap multi-context image representation early on in the network, by means of recursive branching and learnable upsampling, which alleviates the locality-context trade-off concerns inherent in the design of

Im2Im DCNNs.

We believe that several improvements can be made to the RBDN architecture. First, the RBDN architecture could potentially benefit from residual connections, dilated convolutions and possibly other activation functions besides ReLU. Secondly, we used a network of fixed depth across all tasks, which may prove insufficient for complex tasks or suboptimal for simple tasks. The recently proposed Structured Sparsity approach [110] allows networks to simultaneously optimize their hyper-parameters (filter size, depth, local connectivity) in a highly efficient way while training by means of Group Lasso [111] regularization. Thirdly, MSE is known to be an extremely poor [112] loss function for tasks demanding perceptually pleasing image outputs. While the loss function from [108] we used for colorization overcame MSE's limitations, it is specific to the colorization problem. Loss functions based on Adversarial Networks [113] on the other hand can be a generic MSE replacement.

Chapter 4: Furthur Improving Deep Residual Networks by Enhancing Gradient Flow

4.1 Overview

The past few years have witnessed a major paradigm shift in the design of convolutional neural network architectures. Conventional strictly-sequential networks like AlexNet [7], VGG [8] have given way for architectures featuring multiple paths from input to output. Among the many reasons for using multi-path networks (efficiency [114], increased representational capacity [115], feature re-use [116, 117], multiple scales/abstractions [114], ensembling [118], *etc*), one of the most fundamental reasons is to ease optimization when training very deep networks. The appeal of very deep networks stems from the exponential number of regions [119] they can partition the input space into, allowing far richer representations [120] in comparison to wide shallow networks. Highway networks [121], residual networks (ResNets [1]) independently exposed the optimization difficulties in training deep strictly-sequential networks. Highway-nets introduced gated shortcut connections across successive layers, which eased optimization and improved convergence. ResNets on the other hand proposed simple identity shortcuts, whereby every layer

learned a residual function. Despite being a special case of highway networks with identity gates, ResNets displayed substantially improved convergence and performance with fewer parameters and FLOPs. Although ResNets [1] described a particular type of residual building block, the generic residual formulation was applicable to practically any pre-existing architecture. Simply adding identity shortcuts across multiple layers, enabled training deeper networks with improved convergence and performance, even for multi-path architectures like Inception-Nets [115] (which gave way to Inception-ResNet [122]).

After the original ResNet paper (ResNet-v1 [1]), He *et al* proposed a modified *pre-activation* architecture (ResNet-v2 [2]). Figure 4.1 compares the residual blocks for ResNet-v1/v2. ResNet-v2 proposed a generic alteration to any residual architecture, namely to eliminate any non-linearities (like ReLU) after adding the identity shortcuts, instead allowing the residual functions to exclusively handle the non-linear transformations. The resulting network featured a path devoid of non-linearities connecting any two layers, which He *et al* claimed ensured better gradient flow across the network. ResNet-v2 based architectures displayed faster convergence and superior classification performance over their equivalent ResNet-v1 counterparts in CIFAR10/100 for networks of every depth. However, these results on CIFAR surprisingly didn't carry over to the larger ImageNet [6] dataset, with ResNet-v2 outperforming ResNet-v1 only at very high network depths ($\geq$ 152 layers [2]). The same trend was also observed with subsequent state-of-the-art residual architectures such as Wide-ResNets [123], ResNeXts [9], SE-Nets [12]. Classification performance on ImageNet at low computational complexity is often one the most

important criteria for selecting architectures used in modern detectors. As a result, state-of-the-art detectors such as Faster-RCNN [124] use v1-style ResNet-50/101 as their backbone. This overall trend is often unknown to new practitioners, since none of these papers [2, 9, 12, 123, 124] discuss it in detail. Furthermore, the fact that these approaches reverted to v1-style blocks for ImageNet experiments is often mentioned rather obscurely as a minor implementation detail. While He *et al*'s theoretical analysis in [2] showed the benefit of identity shortcuts from an optimization perspective, the benefit of v2-style networks over v1-style networks has only ever been shown empirically on CIFAR10/100. Overall, the reason for the incongruity in performance of v1/v2-style blocks in CIFAR/ImageNet has never been analyzed in any existing work to the best of our knowledge.

Our contributions are two-fold. We first theoretically analyze the differences between v1/v2-style residual networks, in terms of the composition of gradients received by different layers. The parameter updates for every layer in v2-style networks have been shown to predominantly depend on gradients flowing through paths of low *effective depth* [125]. In contrast, our analysis reveals that parameter updates to early layers in v1-style networks depend on a diverse composition of gradients from effectively deeper paths, *i.e.* paths that pass predominantly through residual functions. The gradient composition also varies dynamically over the course of training depending on the indices corresponding to non-zero ReLU outputs at each layer. Our second contribution focuses on the stage-transition residual blocks which typically perform downsampling and increase the number of channels. Standard residual architectures use projection shortcuts for such blocks, since they connect

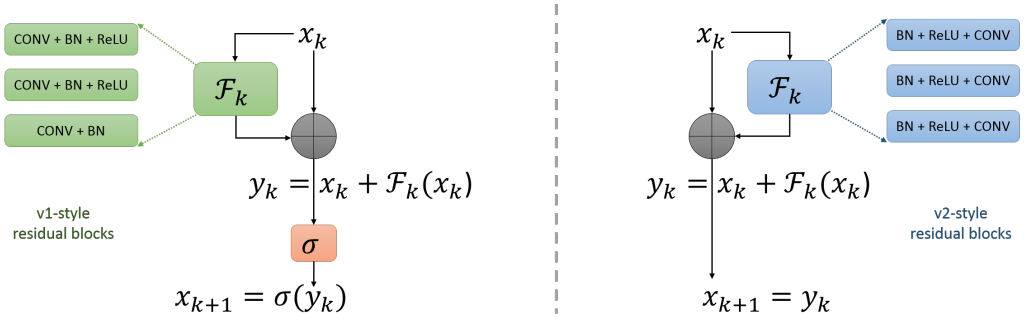

Figure 4.1: **Left:** Original (v1-style [1]) residual block, **Right:** Pre-activation (v2-style [2]) residual block. Note that although [1, 2] describe a particular form of residual function $\mathcal{F}_k$, they can be arbitrary for a generic residual network. The primary difference in v2-style blocks is the absence of an activation function σ after adding shortcuts.

feature maps of different shapes/sizes. While there are only 3-4 projection shortcuts in most networks, we argue that they can significantly throttle gradient flow. We initially try a simple alternative to projection shortcuts in downsampling blocks: merely average-pooling the identity shortcut and concatenating it with the downsampled residual block output (figure 4.2a). This simple modification consistently improves both convergence and the overall performance on CIFAR10/100 for a wide range of residual architectures (table 4.1). Finally, we propose a parameter-free *dense reshape* shortcut(figure 4.3) as an alternative to projection shortcuts or avg-pool+concat in downsampling blocks. Using *dense reshape* shortcuts reduces the top-1 error on ImageNet by 0.5%-1.2% (table 4.4) for ResNets [1], ResNeXts [9] and SE-ResNeXts [12] at depths 50, 101, while preserving the number of FLOPs.

4.2 Related Work

Several investigative studies have been conducted in recent literature to analyze the theoretical properties of residual networks. [118, 125, 126] proposed viewing residual networks as ensembles of shallow networks. Veit *et al* [125] showed that deleting or even shuffling layers in a pre-trained residual network had a surprisingly minimal impact on performance, which inspired several subsequent works. Layer-wise dropout [126] was proposed as a regularizer, which was used extensively in FractalNets [127]. Multi-residual networks [118] were proposed to better exploit the ensembling properties. Greff *et al* [128] showed that unlike strictly-sequential networks which learn increasingly abstract features at each layer, residual networks instead perform iterative unrolled estimation, resulting in successive layers refining estimates of the same features. Balduzzi *et al* [129] presented the problem of *shattered gradients* in deep strictly-sequential neural networks, wherein gradients in deeper layers tend to become increasingly uncorrelated and start resembling white noise. They showed that residual networks mitigate this problem as a consequence of identity skip connections. Orhan *et al* [130, 131] showed that skip connections break symmetries [130] and eliminate singularities [131] in the loss landscape.

The basis for a lot of the recent theoretical results on residual networks is the *unraveled view* by Veit *et al* [125], which highlights multiple *optional* paths of varying depths from input to output. The unraveled view assumes that any intermediate layer output $\boldsymbol{x}_k$ can be written as a sum of multiple functions of any

preceding layer output $\boldsymbol{x}_0$ along with the identity as follows:[1]

$$\boldsymbol{x}_k = \boldsymbol{x}_0 + \mathcal{F}_0\left(\boldsymbol{x}_0\right) + \mathcal{F}_1\left(\boldsymbol{x}_0 + \mathcal{F}_0\left(\boldsymbol{x}_0\right)\right) + \ldots = \boldsymbol{x}_0 + \textstyle\sum_{i=0}^{K-1} \bar{\mathcal{F}}_i\left(\boldsymbol{x}_0\right) \quad \text{where } \bar{\mathcal{F}}_i\left(\boldsymbol{x}_0\right) = \mathcal{F}_i\left(\boldsymbol{x}_i\right). \tag{4.1}$$

Equation 4.1 significantly simplifies further analysis and is critical for a lot of the aforementioned theoretical papers on residual networks. However, it is exclusive to v2-style [2] networks. In v1-style [1] networks, due to a non-linear activation σ (typically ReLU) following the addition of every shortcut, we instead get:

$$\boldsymbol{x}_k = \sigma\left(\bar{\mathcal{F}}_{K-1}(\boldsymbol{x}_0) + \sigma\left(\bar{\mathcal{F}}_{K-2}(\boldsymbol{x}_0) + \sigma\left(\ldots\sigma\left(\bar{\mathcal{F}}_1(\boldsymbol{x}_0) + \sigma\left(\bar{\mathcal{F}}_0(\boldsymbol{x}_0) + \boldsymbol{x}_0\right)\right)\ldots\right)\right)\right) \tag{4.2}$$

The expression for $\boldsymbol{x}_k$ in equation 4.2 no longer contains the additive identity term $\boldsymbol{x}_0$, nor can it be split into a sum of multiple functions of $\boldsymbol{x}_0$, due to the non-linearity of σ. The unraveled view breaks completely and we can't describe v1-style networks as ensembles in the sense described by Veit *et al.*

Nevertheless, motivated by the empirical superiority of v1-style networks on ImageNet as well as their widespread practical usage in state-of-the-art detectors, we analyze them in detail in section 4.3.

4.3 Gradient Flow in v1-style residual networks

Residual networks consist of a modular chain of residual building blocks. The k^{th} block with input $\boldsymbol{x}_k$ and output $\boldsymbol{x}_{k+1}$ is generally expressed as

$$\boldsymbol{y}_k = h_k(\boldsymbol{x}_k) + \mathcal{F}_k(\boldsymbol{x}_k), \quad \boldsymbol{x}_{k+1} = \sigma(\boldsymbol{y}_k) \tag{4.3}$$

[1] assuming there is no downsampling layer between $\boldsymbol{x}_0$ and $\boldsymbol{x}_k$.

where $\mathcal{F}_k(\boldsymbol{x}) = \mathcal{F}(\boldsymbol{x}, \boldsymbol{\theta}_k)$ can be an arbitrary non-linear function parameterized by weights $\boldsymbol{\theta}_k$. When $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k+1}$ have the same feature map-size, the function $h_k()$ is identity for both v1/v2 variants. For stage-transition blocks, $h_k()$ is a projection (typically a 1x1 convolution) whose output matches the dimensions of $\mathcal{F}_k()$. The primary difference between v1-style [1] and v2-style [2] ResNets is the function $\sigma()$. For v1-style ResNets, σ is a rectifier non-linearity (ReLU), while for v2-style ResNets, σ is identity.

We first establish some preliminary notations/results, which would help us succinctly analyze the gradient flow in v1-style networks.

4.3.1 Preliminaries

- For any matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, let $\boldsymbol{r}_i(\boldsymbol{A}) \in \mathbb{R}^n$, $\boldsymbol{c}_j(\boldsymbol{A}) \in \mathbb{R}^m$, $e_{i,j}(\boldsymbol{A}) \in \mathbb{R}$ respectively denote the i^{th} row, j^{th} column and the element at (i, j).

- The product of K matrices $\boldsymbol{A}_1 \boldsymbol{A}_2 \cdots \boldsymbol{A}_K$ ($\boldsymbol{A}_t \in \mathbb{R}^{n_t \times n_{t+1}}$) can be expressed as a summation of outer products between columns of $\boldsymbol{A}_1$ and rows of $\boldsymbol{A}_K$:

$$\prod_{t=1}^{K} \boldsymbol{A}_t = \sum_{i_1=1}^{n_2} \cdots \sum_{i_{K-1}=1}^{n_K} \prod_{t=2}^{K-1} e_{i_{t-1}, i_t}(\boldsymbol{A}_t) \boldsymbol{c}_{i_1}(\boldsymbol{A}_1) \boldsymbol{r}_{i_{K-1}}^T(\boldsymbol{A}_K) \tag{4.4}$$

  The form in equation 4.4 will be particularly useful in isolating the effect of ReLU ($\sigma()$ from equation 4.3) in v1-style resnets, which induces structured sparsity in each gradient term by zeroing out specific rows/columns.

- For a vector $\boldsymbol{x} \in \mathbb{R}^n$, let $x^+ \subseteq \{1, \ldots, n\}$ denote the set of indices corresponding to +ve elements in $\boldsymbol{x}$.

- For row indices $H \subseteq \{1, \ldots, m\}$ and column indices $W \subseteq \{1, \ldots, n\}$, define $[\![\boldsymbol{A}]\!]_W^H \in \mathbb{R}^{m \times n}$ with elements

$$e_{i,j}([\![\boldsymbol{A}]\!]_W^H) = \begin{cases} e_{i,j}(\boldsymbol{A}) & \text{if } i \in H \text{ and } j \in W \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

  For convenience, we drop super-script H or sub-script W when no rows or columns are zeroed respectively. Note that $[\![\boldsymbol{A}]\!]_W^H = [\![\boldsymbol{I}]\!]^H \boldsymbol{A} [\![\boldsymbol{I}]\!]^W$.

- The derivative of ReLU:

$$\frac{\partial \sigma(\boldsymbol{x})}{\partial \boldsymbol{x}} = [\![\boldsymbol{I}]\!]^{x^+} \quad \text{where } \boldsymbol{I} \text{ is the } n \times n \text{ identity matrix.} \tag{4.6}$$

- For indices $H_t \in \{1, \ldots, n_t\}, W_t \in \{1, \ldots, n_{t+1}\}$, we can express the product $[\![\boldsymbol{A}_1]\!]_{W_1}^{H_1} [\![\boldsymbol{A}_2]\!]_{W_2}^{H_2} \cdots [\![\boldsymbol{A}_K]\!]_{W_K}^{H_K}$ in the form of equation 4.4:

$$\begin{aligned} \prod_{t=1}^{K} [\![\boldsymbol{A}_t]\!]_{W_t}^{H_t} &= \sum_{i_1=1}^{n_2} \cdots \sum_{i_{K-1}=1}^{n_K} \prod_{t=2}^{K-1} e_{i_{t-1},i_t}([\![\boldsymbol{A}_t]\!]_{W_t}^{H_t}) \boldsymbol{c}_{i_1}([\![\boldsymbol{A}_1]\!]_{W_1}^{H_1}) \boldsymbol{r}_{i_{K-1}}^T([\![\boldsymbol{A}_K]\!]_{W_K}^{H_K}) \\ &= \sum_{\substack{i_1 \in \\ W_1 \cap H_2}} \cdots \sum_{\substack{i_j \in \\ W_j \cap H_{j+1}}} \cdots \sum_{\substack{i_{K-1} \in \\ W_{K-1} \cap H_K}} \prod_{t=2}^{K-1} e_{i_{t-1},i_t}(\boldsymbol{A}_t) \left[\!\!\left[\boldsymbol{c}_{i_1}(\boldsymbol{A}_1) \boldsymbol{r}_{i_{K-1}}^T(\boldsymbol{A}_K) \right]\!\!\right]_{W_K}^{H_1} \end{aligned} \tag{4.7}$$

- Equation 4.7 is composed of $\prod_{t=1}^{K-1} |W_t \cap H_{t+1}|$ sums of outer product terms, each of which has zero elements for rows with indices in $\widetilde{H_1}$ and columns with indices in $\widetilde{W_K}$ ($\widetilde{X}$ denotes complement of X).

4.3.2 v1-style ResNets

By applying the chain-rule of differentiation to equation 4.3, the gradient with respect to the loss function ℓ is back-propagated as

$$\frac{\partial \ell}{\partial \boldsymbol{x}_k} = \frac{\partial \ell}{\partial \boldsymbol{x}_{k+1}} \frac{\partial \boldsymbol{x}_{k+1}}{\partial \boldsymbol{y}_k} \frac{\partial \boldsymbol{y}_k}{\partial \boldsymbol{x}_k} = \frac{\partial \ell}{\partial \boldsymbol{x}_{k+1}} \frac{\partial \sigma(\boldsymbol{y}_k)}{\partial \boldsymbol{y}_k} \left(\frac{\partial h(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_k} + \frac{\partial \mathcal{F}_k(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_k} \right). \tag{4.8}$$

We first consider a sequence of L residual blocks within a stage corresponding to activations $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_L$. For such blocks, we can set $h_k(\boldsymbol{x}_k) = \boldsymbol{x}_k$. Let $\frac{\partial \mathcal{F}_k(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_k} = \boldsymbol{G}_k$. From equation 4.6, we obtain

$$\begin{aligned} &\frac{\partial \ell}{\partial \boldsymbol{x}_0} = \frac{\partial \ell}{\partial \boldsymbol{x}_1} [\![\boldsymbol{I}]\!]^{y_0^+} (\boldsymbol{I} + \boldsymbol{G}_0) = \frac{\partial \ell}{\partial \boldsymbol{x}_1} \left([\![\boldsymbol{I}]\!]^{y_0^+} + [\![\boldsymbol{G}_0]\!]^{y_0^+} \right) = \\ &\frac{\partial \ell}{\partial \boldsymbol{x}_2} \left([\![\boldsymbol{I}]\!]^{y_1^+} + [\![\boldsymbol{G}_1]\!]^{y_1^+} \right) \left([\![\boldsymbol{I}]\!]^{y_0^+} + [\![\boldsymbol{G}_0]\!]^{y_0^+} \right) = \frac{\partial \ell}{\partial \boldsymbol{x}_L} \prod_{k=1}^{L} \left([\![\boldsymbol{I}]\!]^{y_{L-k}^+} + [\![\boldsymbol{G}_{L-k}]\!]^{y_{L-k}^+} \right) \end{aligned} \tag{4.9}$$

By expanding the product in equation 4.9,

$$\begin{aligned} &\prod_{k=1}^{L} \left([\![\boldsymbol{I}]\!]^{y_{L-k}^+} + [\![\boldsymbol{G}_{L-k}]\!]^{y_{L-k}^+} \right) = [\![\boldsymbol{I}]\!]^{\bigcap_{k=0}^{L-1} y_k^+} \text{ (identity term)} \\ &+ \sum_{i=0}^{L-1} [\![\boldsymbol{G}_i]\!]^{\bigcap_{k=i}^{L-1} y_k^+}_{\bigcap_{k=0}^{i-1} y_k^+} \text{ (order-1 terms)} \; + \sum_{i=1}^{L-1} \sum_{j=0}^{i-1} [\![\boldsymbol{G}_i]\!]^{\bigcap_{k=i}^{L-1} y_k^+} [\![\boldsymbol{G}_j]\!]^{\bigcap_{k=j}^{i-1} y_k^+}_{\bigcap_{k=0}^{j-1} y_k^+} \text{ (order-2 terms)} \\ &+ \cdots \sum_{\substack{i_0, \ldots, i_{K-1} = 0 \\ i_{K-1} > \cdots > i_0}}^{L-1} [\![\boldsymbol{G}_{i_{K-1}}]\!]^{\bigcap_{k=i_{K-1}}^{L-1} y_k^+} \prod_{j=2}^{K-1} [\![\boldsymbol{G}_{i_{K-j}}]\!]^{\bigcap_{k=i_{K-j}}^{i_{K-j+1}-1} y_k^+} [\![\boldsymbol{G}_{i_0}]\!]^{\bigcap_{k=i_0}^{i_1-1} y_k^+}_{\bigcap_{k=0}^{i_0-1} y_k^+} \text{ (order-}K\text{ terms)} \\ &+ \cdots \prod_{k=1}^{L} [\![\boldsymbol{G}_{L-k}]\!]^{y_{L-k}^+} \text{ (order-}L\text{ term)} \end{aligned} \tag{4.10}$$

The general form of an order-K term in equation 4.10 can be further expanded as a sum of outer products using equation 4.7. For $1 < K < L$, if we fix indices

$i_0, i_1, \ldots, i_{K-1}$ (with $i_{j+1} > i_j$), the product

$$[\![\boldsymbol{G}_{i_{K-1}}]\!]^{\bigcap_{k=i_{K-1}}^{L-1} y_k^+} \prod_{j=2}^{K-1} [\![\boldsymbol{G}_{i_{K-j}}]\!]^{\bigcap_{k=i_{K-j}}^{i_{K-j+1}-1} y_k^+} [\![\boldsymbol{G}_{i_0}]\!]_{\bigcap_{k=0}^{i_0-1} y_k^+}^{\bigcap_{k=i_0}^{i_1-1} y_k^+}$$

$$= \sum_{\substack{x_t \in \bigcap_{k=i_{K-t-1}}^{i_{K-t}-1} y_k^+ \\ t \in \{1,\ldots,K-1\}}} \prod_{t=2}^{K-1} e_{x_{t-1},x_t}\left(\boldsymbol{G}_{i_{K-t}}\right) \left[\!\!\left[\boldsymbol{c}_{x_1}\left(\boldsymbol{G}_{i_{K-1}}\right) \boldsymbol{r}^T_{x_{K-1}}\left(\boldsymbol{G}_{i_0}\right) \right]\!\!\right]_{\bigcap_{k=0}^{i_0-1} y_k^+}^{\bigcap_{k=i_{K-1}}^{L-1} y_k^+} \tag{4.11}$$

Equations 4.10, 4.11 reveal some key insights:

- The sparsity of the identity term as well as first order terms depends on all L activations $(y_0^+, \ldots, y_{L-1}^+)$ and their influence vanishes for large L.
- The sparsity (number of zeroed rows/columns) in each order-K term corresponding to a sequence of residual blocks $i_0 < i_1 \ldots < i_{K-1}$ depends only on the first and last block, *i.e.* i_0 and i_{K-1}.
  - For an order-K term to have a non-zero r^{th} row, each of the $(L - i_{K-1})$ activations $(\boldsymbol{y}_{i_{K-1}}, \ldots, \boldsymbol{y}_{L-1})$ must have a positive r^{th} element.
  - For an order-K term to have a non-zero c^{th} column, each of the i_0 activations $(\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i_0-1})$ must have a positive c^{th} element.
- The intermediate blocks in an order-K term $(i_1, \ldots, i_{K-2})$ influence the total number of terms in its outer-product expansion: $\prod_{t=1}^{K-1} \left| \bigcap_{k=i_{K-t-1}}^{i_{K-t}-1} y_k^+ \right|$. Note that if $\bigcap_{k=i_{K-t-1}}^{i_{K-t}-1} y_k^+ = \emptyset$ for some t, then the entire order-K term vanishes. The likelihood of this happening is higher for those terms wherein 2 successive blocks $i_{K-t-1},\ i_{K-t}$ are fairly distant. For example, a term $\boldsymbol{G}_{20}\boldsymbol{G}_3\boldsymbol{G}_2\boldsymbol{G}_1\boldsymbol{G}_0$ has a higher likelihood of vanishing in comparison to say $\boldsymbol{G}_{20}\boldsymbol{G}_{15}\boldsymbol{G}_{10}\boldsymbol{G}_5\boldsymbol{G}_0$.

- The gradient composition at $\boldsymbol{x}_0$ is dynamic as training progresses. A change in y_k^+ (for any $0 \leq k < L$) alters the relative influence of every residual block.

In summary, as the number of residual blocks L increases, the gradient at $\boldsymbol{x}_0$ is increasingly dominated by higher order terms whose relative influence is dynamic over the course of training. Order-K terms with $K \ll L$ either become exceedingly sparse or vanish entirely.

4.3.3 Differences between v1 and v2-style ResNets

We can revert to analyzing v2-style networks by simply removing all y_k^+ terms in equations 4.10, 4.11. Both v1, v2 variants have a total of 2^L gradient terms in equation 4.10, with $\binom{L}{K}$ order-K terms. The key difference is that in v2-style networks, *all* 2^L terms are simply added together, which can undesirably cause a small fraction of terms with high norms to dominate and overshadow all other terms. Veit *et al* empirically showed that in practice, the gradient flowing through each residual function almost always has a lower norm in comparison to gradients flowing through the shortcut connection [125], *i.e.* $\|\frac{\partial \ell}{\partial \boldsymbol{x}_k}\| < \|\frac{\partial \ell}{\partial \boldsymbol{x}_k}\boldsymbol{G}_k\|$. This is what primarily causes v2-style networks to have a low *effective depth*. While the number of order-K terms $= \binom{L}{K}$ which follows a binomial distribution increases with K up to $K = \frac{L}{2}$, the average gradient magnitude m_K over random paths of length K drops exponentially with K. Viet *et al* loosely defined effective depth as the value of K which empirically maximized $\binom{L}{K} m_K$. (They found this value to be much lower than $\frac{L}{2}$.)

v1-style networks on the other hand can reduce the dominating influence of lower-order terms by introducing a higher level of sparsity for lower values of K and can additionally cause certain low-order terms to vanish entirely. Furthermore, each of the different order-K terms that don't vanish operate on a potentially different subset of $\frac{\partial \ell}{\partial \boldsymbol{x}_L}$ (governed by activations $\{\boldsymbol{y}_{i_{K-1}}, \ldots, \boldsymbol{y}_{L-1}\}$) and contribute to a potentially different subset of $\frac{\partial \ell}{\partial \boldsymbol{x}_0}$ (governed by activations $\{\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i_0-1}\}$). This makes the composition of gradients *diverse*, *i.e.* the relative influence of each of the 2^L terms can potentially be very different for each index of $\frac{\partial \ell}{\partial \boldsymbol{x}_0}$.

We surmise that the superiority of v2-style architectures for the CIFAR classification task is primarily due to the relatively small training set in comparison to ImageNet. Since CIFAR only has 50000 training samples, ensembles of shallow networks perhaps generalize better to the validation set. In contrast, for ImageNet which has over 1.2 million training images with 1000 classes, the potentially higher effective depth offered by v1-style networks coupled with a dynamic and diverse gradient composition at each layer, gives them an edge over v2-style networks despite losing out on the elegant ensembling properties that v2-style networks posses. For the remainder of the chapter, we use v2-style networks for all CIFAR experiments and v1-style networks for all ImageNet experiments, unless specified otherwise.

4.4 Downsampling shortcuts

In equation 4.9, we considered L residual blocks that were part of the same stage and dealt with feature maps of the same size. If we instead consider the j^{th}

block (for some $0 \leq j < L$) to use a projection shortcut $h_j()$ for downsampling (with derivative $\boldsymbol{H}_j = \frac{\partial h_j(\boldsymbol{x}_j)}{\partial \boldsymbol{x}_j}$), we can rewrite equation 4.9 as:

$$\frac{\partial \ell}{\partial \boldsymbol{x}_0} = \frac{\partial \ell}{\partial \boldsymbol{x}_L} \prod_{k=1}^{L-j+1} \Big([\![\boldsymbol{I}]\!]^{y^+_{L-k}} + [\![\boldsymbol{G}_{L-k}]\!]^{y^+_{L-k}} \Big) \Big([\![\boldsymbol{H}_j]\!]^{y^+_j} + [\![\boldsymbol{G}_j]\!]^{y^+_j} \Big) \prod_{k=1}^{j} \Big([\![\boldsymbol{I}]\!]^{y^+_{j-k}} + [\![\boldsymbol{G}_{j-k}]\!]^{y^+_{j-k}} \Big) \quad (4.12)$$

When expanding the product, it can be seen that every single gradient term will either have to contain $[\![\boldsymbol{H}_j]\!]^{y^+_j}$ or $[\![\boldsymbol{G}_j]\!]^{y^+_j}$. If both $[\![\boldsymbol{H}_j]\!]^{y^+_j}$ and $[\![\boldsymbol{G}_j]\!]^{y^+_j}$ have vanishing gradients at some training iteration, the gradients to all earlier layers $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{j-1}$ will be throttled.

While this applies to projection shortcuts in general, downsampling projections present another issue in practice. Most of the standard residual architectures use 1×1 stride-2 convolutions for projections, which effectively discard 75% of their input. As a result, the gradients will only be back-propagated to 25% of the input from the projection path, forcing the bulk of the gradients to flow through the residual functions. This issue can be significantly compounded with multiple downsampling shortcuts. With 3 downsampling shortcuts (common for architectures used in ImageNet), $98.44\%({}^{63}/_{64})$ of the gradients in the final stage will be forced to flow through 3 downsampling residual functions before reaching layers in the first stage.

4.4.1 Average-Pool + Concat as an alternative

We first experiment with a simple alternative to projection shortcuts used for downsampling. The identity shortcut is average-pooled and concatenated with the residual block output as shown in figure 4.2a. The number of output channels in the corresponding residual function is reduced to ensure that we obtain the desired

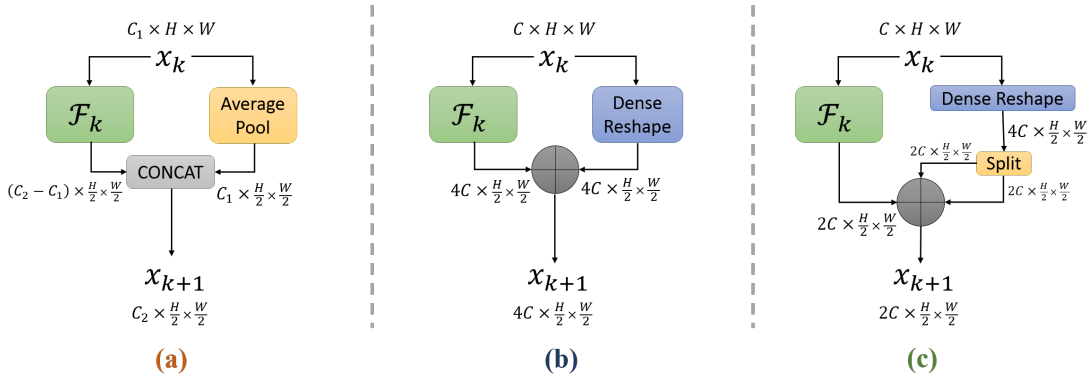

Figure 4.2: Exploring different types of downsampling shortcuts **(a):** Average-Pool + Concat, **(b,c):** *Dense Reshape* shortcuts.

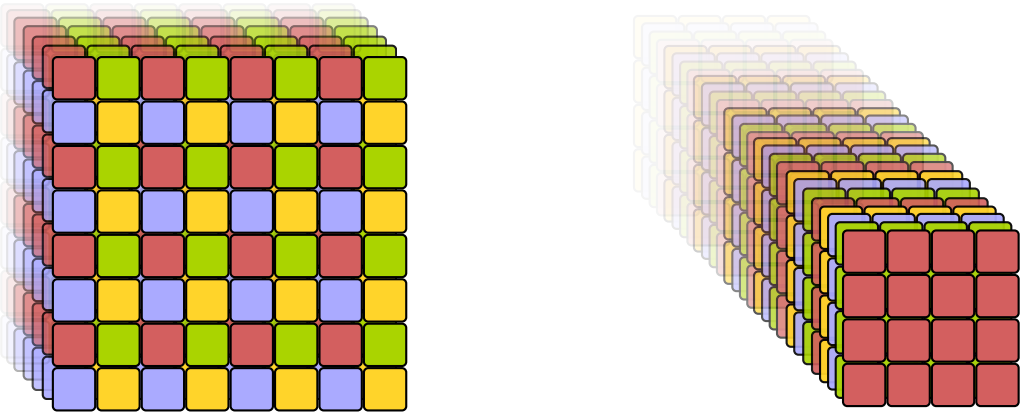

Figure 4.3: *Dense reshape* operation for $r = 2$ (best seen in color). **Left:** $(C \times 2H \times 2W)$ input, **Right:** $(4C \times H \times W)$ output. Every $(2H \times 2W)$ input slice is transformed into a $(4 \times H \times W)$ tensor and concatenated. The operation is cheap, reversible, preserves spatial structure and gradient flow.

Table 4.1: Top-1 test classification error on CIFAR10/100 test set. The experiment involves replacing downsampling projections in baselines with average-pooled identity shortcuts that are concatenated with downsampled residual block outputs. All networks are trained with standard data-augmentations: random crops + flips on the CIFAR10/100 training set (ZCA normalization is not used).

Network	CIFAR10		CIFAR100	
	projections	avg-pool + concat	projections	avg-pool + concat
ResNet-110 [2]	6.37%	5.45%	25.92%	24.30%
ResNet-164 [2]	5.46%	4.81%	24.33%	22.58%
ResNet-1001 [2]	4.92%	-	22.71%	-
WRN-28-10-dropout [123]	3.89%	3.53%	18.85%	18.15%
ResNeXt-29-8-64 [9]	3.65%	3.47%	17.77%	17.23%
ResNeXt-29-16-64 [9]	3.58%	**3.25%**	17.31%	**16.71%**

Table 4.2: Training protocol for *all* ImageNet experiments.

batch-size: 256, **crop-size:** 224×224, **GPUs:** 4 Quadro P6000 (24 GB memory each)

initialization: Xavier, **base LR:** 0.1, dropped by factor 0.1 at epochs $\{60, 90, 120\}$.

optimizer: Nesterov (*momentum:* 0.9, *weight decay:* 0.0001), **framework:** MXNet (commit 83078d7)

Epoch	**Data Augmentations**
0-120	random crops/flips, random resize (shorter side $\in [256, 480]$), random aspect ratio $\in [0.75, 1.25]$, random color (HSL $\in \pm 40\%$)
120-135	Only random crops/flips with shorter side resized to 256.

Table 4.3: Comparing FLOPs used by projection shortcuts to residual blocks in ResNet/ResNeXt. With the exception of stage transition blocks, the FLOPs in columns $2, 3$ are the same for all residual blocks in ResNets/ResNeXts respectively, since a $2\times$ reduction in each spatial dimension is followed by a $2\times$ increase in number of channels.

	Projection Shortcuts	ResNet [1] residual block	ResNeXt [9] residual block
FLOPs	56 x 56 x 64 x 256 28 x 28 x 256 x 512 14 x 14 x 512 x 1024 7 x 7 x 1024 x 2048	56 x 56 x 256 x 64 56 x 56 x 3 x 3 x 64 x 64 56 x 56 x 64 x 256	56 x 56 x 256 x 128 56 x 56 x 3 x 3 x 128 x 128/32 56 x 56 x 128 x 256
Total	56 x 56 x **28** x 64 x 64	56 x 56 x **17** x 64 x 64	56 x 56 x **17.125** x 64 x 64

number of channels after concatenation. We try this simple modification on CIFAR10/100 (table 4.1) with a host of residual architectures such as ResNets [2], Wide-ResNets [123] and ResNeXts [9]. In every case, we obtain a significant improvement over the baseline despite using fewer FLOPs. Compared to baseline architectures, we save FLOPs by eliminating projections and by reducing the number of output channels in the residual function. From table 4.1, we can also see that a 164-layer ResNet with avg-pool+concat outperforms a 1001-layer baseline ResNet on both CIFAR10/100.

The reason for the success of this simple modification can be attributed to at least 50% of the gradients flowing unaltered through the average-pooled shortcuts. However, this alternative only partially eliminates the potential problems caused by downsampling projections, since 50% of the gradients are still forced to flow through the residual functions which perform downsampling.

4.4.2 Dense Reshape Shortcuts

We finally propose *dense reshape* shortcuts (figure 4.3) as an alternative to projection shortcuts or average-pool+concat for downsampling blocks. These shortcuts **(a)** discard minimal information, **(b)** preserve spatial structure of activation, **(c)** are parameter-free, cheap to compute, and are **(d)** *identity-like*, *i.e.* pass/route the gradient unaltered during backward pass.

The *dense reshape* operation in general takes as input a $(C \times H.r \times W.r)$ tensor and produces a $(C.r^2 \times H \times W)$ tensor as output (C, H, W respectively denote the channel, height and width dimensions). An input element at location (c, h, w) is sent to output location (c', h', w') given by:

$$c' = c.r^2 + (h \mod r).r + (w \mod r), \quad h' = \left\lfloor \frac{h}{r} \right\rfloor, \quad w' = \left\lfloor \frac{w}{r} \right\rfloor. \tag{4.13}$$

During backward pass, the gradient at output location (c', h', w') is routed to input location (c, h, w) given by:

$$c = \frac{c'}{r^2}, \quad h = h'.r + \left(\left\lfloor \frac{c'}{r} \right\rfloor \mod r \right), \quad w = w'.r + (c' \mod r). \tag{4.14}$$

Note that the operation does not discard or alter the data in any way, but merely redirects it. Equations 4.13, 4.14 can be implemented very efficiently in parallel using only integer arithmetic. In GPUs, the operation is practically instantaneous within a CUDA loop, since no floating-point operations are performed on the data.

When the number of output channels in a downsampling block is $4\times$ its input (*ex:* first residual block in ImageNet architectures go from 64 to 256 channels), the dense reshaped output with $r = 2$ is simply added to the residual function output.

When the number of output channels is $2\times$ its input, the dense reshaped output is equally split along the channel dimension and both splits are added to the residual function output (as shown in figure 4.2c).

The *dense reshape* operation with scale r can be seen as a special case of *im2col* [132, 133] with a $r \times r$ stride-r kernel and is equivalent to the *space-to-depth* module in the Tensorflow [134] framework. While *space-to-depth* is not commonly used, its inverse-operation: *depth-to-space* is very popular in *super-resolution* networks [135–137] for upsampling activations without producing checkerboard artifacts [138].

4.5 Experiments

We experiment with our proposed *dense reshape* shortcuts on the ImageNet classification benchmark on popular residual architectures like ResNets [1], ResNeXts [9] and SE-ResNeXts [12] at depths 50 and 101. ImageNet [6] consists of ~ 1.2 million training images with 1000 classes and $50,000$ validation images. In order to make a fair comparison, we retrain all the baseline networks from scratch and use an *identical* training setup with identical data-augmentations for every single network as described in table 4.2. The type of data-augmentations used can significantly influence the final top-1 validation-accuracy of the final model for the same architecture (often by $0.5-1.5\%$). Our major reason for retraining the baselines is the prominent differences in data-augmentations used in different residual networks. For instance, ResNeXts [9] additionally used PCA-based lighting noise [7] while SE-Nets [12] addi-

Table 4.4: Comparing all 3 downsampling shortcuts: **Projection(P), Avg-Pool+Concat(AP), Dense-Reshape(DR)** on ImageNet. For models with **AP, DR** shortcuts, we add 2 additional residual blocks at stage 3 to preserve complexity with respect to an equivalent model with **P** shortcuts.

Top-1 (Top-5) single 224 x 224 center-crop validation error on ImageNet for v1-style ResNets						
Shortcut	Projection (P)	Avg-Pool+Concat (AP)	Dense Reshape (DR)	**P - AP**	**AP - DR**	**P - DR**
ResNet-50	24.75 (7.53)	24.37 (7.32)	23.56 (6.67)	0.38 (0.21)	0.81 (0.65)	1.19 (0.86)
ResNeXt-50	23.04 (6.48)	22.89 (6.41)	22.11 (6.04)	0.15 (0.07)	0.78 (0.37)	0.93 (0.44)
SE-ResNeXt-50	22.35 (6.23)	22.18 (6.17)	21.56 (5.85)	0.17 (0.06)	0.62 (0.32)	0.79 (0.38)
ResNet-101	22.80 (6.36)	22.69 (6.39)	22.37 (6.21)	0.11 (-.03)	0.32 (0.18)	0.43 (0.15)
ResNeXt-101	21.92 (5.86)	21.82 (5.88)	21.38 (5.73)	0.10 (-.02)	0.44 (0.15)	0.54 (0.13)

tionally used random rotations, pixel jittering and more aggressive scale/aspect-ratio augmentations compared to the augmentations we used in table 4.2.

4.5.1 Reallocating FLOPs originally used for projections

The bottleneck residual architectures used in ImageNet devote a considerable number of FLOPs for the projection shortcuts due to the relatively large number of channels at each stage in comparison to non-bottleneck architectures. Table 4.3 compares the total FLOPs used by projection shortcuts to FLOPs used by each residual block in ResNets/ResNeXts. Since we replace projections with our *dense reshape* shortcuts that have practically no FLOPs, we add 2 additional residual blocks to roughly preserve the computational complexity of the network. The 2 additional blocks are added at stage 3 of the network, that deals with activations of size $(14 \times 14 \times 1024)$, in order to additionally preserve the number of parameters and GPU memory consumption with respect to the baseline network.

Table 4.5: Comparing **v1-style** [1] & **v2-style** [2] ResNets with **projection(P)** or **dense-reshape(DR)** downsampling shortcuts on ImageNet. Again, all **DR** models have 2 additional residual blocks at stage 3 to balance FLOPs.

	Top-1 (Top-5) single 224 x 224 center-crop validation error on ImageNet					
Style (Shortcut)	**v2 (P)**	**v2 (DR)**	**v2(P) - v1(P)**	**v2(DR) - v1(DR)**	**v2(P) - v2(DR)**	**v1(P) - v2(DR)**
ResNet-50	25.14 (7.64)	24.44 (7.41)	0.39 (0.11)	0.88 (0.74)	0.70 (0.23)	0.31 (0.12)
ResNeXt-50	23.39 (6.56)	22.78 (6.32)	0.35 (0.08)	0.67 (0.28)	0.61 (0.24)	0.26 (0.16)
SE-ResNeXt-50	22.52 (6.31)	22.25 (6.17)	0.17 (0.08)	0.69 (0.32)	0.27 (0.14)	0.10 (0.06)

4.6 Results

4.6.1 Comparing Different Downsampling Shortcuts

Table 4.4 shows the top-1, top-5 validation error on ImageNet with popular residual architectures for all 3 downsampling shortcuts: **Projection(P), Avg-Pool+Concat(AP), Dense-Reshape(DR)** on ImageNet. Using *average-pool + concat* only gives marginal improvement (Column **5**) over *projection* shortcuts, while using *dense reshape* shortcuts consistently improves performance(Column **6**, Column **7**) for all architectures, with the improvements being much more significant at depth 50.

4.6.2 Comparing v1 vs v2 for both P/DR shortcuts

Table 4.5 compares **v1-style** [1] & **v2-style** [2] ResNets using projection(**P**) or dense-reshape(**DR**) downsampling shortcuts on ImageNet. At roughly the same computational complexity (FLOPs),

Table 4.6: Single 224×224 center-crop validation error for ResNet-50 and ResNet-56 (2 more residual blocks at stage 3) with various types of downsampling shortcuts. For the 3rd row, we use 2 shortcut connections during downsampling: the dense reshape and the projection shortcut, both of which are added to the residual function output.

Model	**Downsampling shortcut type**	**GFLOPs**	**Top-1 (Top-5) validation error**	**Improvement over ResNet-50**	**Improvement over ResNet-56**
ResNet-50	Projection	4.09	24.75 (7.53)	-	-
	Dense-Reshape	3.73	24.10 (7.09)	0.65 (0.44)	0.08 (0.05)
	Both	4.09	23.71 (6.89)	1.04 (0.64)	0.47 (0.25)
ResNet-56	Projection	4.53	24.18 (7.14)	0.57 (0.39)	-
	Dense-Reshape	4.17	23.56 (6.67)	1.19 (0.86)	0.62 (0.47)

- Column 4: **v1** gives marginal improvement over **v2** using **P**-shortcuts.
- Column 5: **v1** gives significant improvement over **v2** using **DR**-shortcuts.
- Column 6: Using **DR** over **P**-shortcuts improves performance for **v2**-ResNets.
- Column 7: **v1** + **P**-shortcuts are slightly better than **v2** + **DR**-shortcuts.
- Overall ImageNet performance of **v1(DR)** > **v1(P)** > **v2(DR)** > **v2(P)**.

4.6.3 The influence of 2 additional residual blocks

The overall improvements after using *dense reshape* shortcuts can arise due to two confounding factors: **(a)** the shortcuts themselves and/or **(b)** the 2 additional residual blocks that are added to preserve complexity. In order to isolate the rel-

ative improvements brought forth separately by **(a)** and **(b)**, we perform further experiments using ResNet-50 in table 4.6:

- We train a ResNet-50 with *dense reshape* shortcuts without the 2 additional residual blocks (row 2 in table 4.6). This network still outperforms the baseline ResNet-50 by 0.64%, despite having 9% fewer FLOPs, which shows that *dense reshape* shortcuts are independently beneficial.

- We train a ResNet-56 (2 additional residual blocks at stage 3) with projection shortcuts for downsampling (row 4 in table 4.6). While this network outperforms baseline ResNet-50, it surprisingly only performs as well as the ResNet-50 with *dense reshape* shortcuts, which has 19% fewer FLOPs.

- We additionally train a ResNet-50 (without the 2 additional residual blocks) that uses both projections and dense-reshape for downsampling, *i.e.* both the shortcuts are added to the residual function output during downsampling (row 3 in table 4.6). This network surprisingly outperforms not only the baseline ResNet-50, but also the ResNet-50 with *dense-reshape* shortcuts and ResNet-56 with projection shortcuts. Its performance is only marginally inferior to ResNet-56 with *dense-reshape* shortcuts. This suggests that projection shortcuts might be useful when used alongside identity shortcuts. If an identity path for gradients already exists, the projection shortcuts can instead behave as an alternate residual function that learns a different transformation of its input.

4.7 Discussion

- *Average-Pool + Concat* (**AP**) shortcuts don't improve results in ImageNet as much as they do in CIFAR. ImageNet has 4 stage-transitions in comparison to CIFAR which only has 2. Unlike *dense reshape* (**DR**), each **AP** shortcut forces half the gradients to flow through the residual function $\mathcal{F}$ as mentioned in section 4.4.1. This effect is more severe in ImageNet, since 93.75%(15/16) of gradients that are received by activations in the first stage, are forced to flow through at least one residual function.

- Since **v1**-style networks can utilize a higher effective depth, they benefit much more from the improved gradient flow provided by **DR** shortcuts in comparison to **v2**, as can be seen by comparing Column **4** and Column **5** in table 4.5.

4.8 Summary

We present a theoretical analysis on post-activation (**v1**-style) [1] residual architectures, which reveals some potential insights on why they outperform pre-activation (**v2**-style) [2] architectures on ImageNet [6] and not on CIFAR. We decouple the influence of post/pre-activation and projection shortcuts and show that unlike projections, post-activation isn't necessarily bad for optimization and *doesn't throttle* gradient flow. While **v1**-style nets lose out on ensembling properties, they enjoy other interesting features like a potentially higher effective depth due to a diverse composition of gradients being forced to flow through maximal residual func-

tions. In practice, **v2** works better for smaller datasets like CIFAR, which benefit from the generalization afforded by ensembles of shallow nets, while **v1** works better for large multi-class datasets like ImageNet which benefit from the greater modeling capacity afforded by higher effective depth.

We additionally highlight the pitfalls of downsampling projections, by showing that even simple alternatives like *average pool + concat* can significantly improve classification performance on CIFAR. Our proposed *dense reshape* shortcuts serve as a natural extension of *identity* shortcuts, which are parameter-free, cheap to compute and preserve spatial structure and gradient flow. Using *dense reshape* shortcuts consistently improves classification performance over both projection shortcuts and *average pool + concat* at the same FLOPs on ImageNet for a wide range of popular **v1/v2** residual architectures like ResNets [1], ResNeXts [9] and SENets [12]. Furthermore, since the performance improvement is much higher at lower depths, *dense reshape* shortcuts can be particularly useful in fast real-time detection-modules [124], which typically utilize low-depth residual networks like ResNet [1]/ResNeXt-50 [9] as their backbone.

Chapter 5: Conclusion

In this thesis, we presented novel approaches based on low-level vision, machine-learning and optimization theory, that can be used off-the-shelf to improve and augment a wide range of pre-existing object recognition pipelines.

In the first part (chapter 2), we proposed a robust non-parametric probabilistic ensemble method KDEMRP, which is particularly well suited for multi-classification tasks with limited data and potential outliers, where end-to-end DCNNs often fail. KDEMRP produces statistically significant improvements over state-of-the-art ensembling techniques for several standard *machine learning* and *computer vision* datasets. KDEMRP works with arbitrary linear/non-linear multi-classifiers and consistently improves results.

In the second part (chapter 3, we focused on the context-vs-locality tradeoff inherent in DCNN architectures designed for image-to-image regression. We presented a generic DCNN for Im2Im regression called RBDN, that not only eliminates the context-vs-locality tradeoff, but can additionally learn the relative influence of context of locality based on task at hand. The single RBDN architecture obtains comparable results to state-of-the-art approaches for diverse tasks like relighting, denoising and colorization. For denoising, we obtain state-of-the-art results using a

single model for blind-denoising which even beats all noise-level specific competitor approaches at high noise levels by ~1db.

In the last part (chapter 4), we compare and contrast gradient flow in v1-style [1] and v2-style [2] ResNets. Our analysis indicates that while v1-style ResNets lose out on ensembling properties, they receive a diverse dynamic composition of gradients from effectively deeper paths in the network. Finally, we propose dense-reshape shortcuts as an alternative to projection shortcuts used in downsampling stages, which preserve gradient flow. Using dense-reshape shortcuts significantly improves performance on ImageNet at the same FLOPs.

Bibliography

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, viii, x, 3, 5, 6, 41, 61, 64, 65, 67, 69, 70, 79, 81, 83, 86, 87, 89

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016. 1, viii, x, 6, 65, 66, 67, 69, 70, 78, 79, 83, 86, 89

[3] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *arXiv preprint arXiv:1608.03981*, 2016. vii, x, 45, 58, 59, 61

[4] Ralph Gross, Iain Matthews, Jeffrey Cohn, Takeo Kanade, and Simon Baker. Multi-pie. *Image Vision Comput.*, 28(5):807–813, May 2010. x, 54, 55, 56, 57

[5] Brendan F Klare, Ben Klein, Emma Taborsky, Austin Blanton, Jordan Cheney, Kristen Allen, Patrick Grother, Alan Mah, Mark Burge, and Anil K Jain. Pushing the frontiers of unconstrained face detection and recognition: Iarpa janus benchmark a. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1931–1939. IEEE, 2015. x, 57, 61

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. 3, 65, 81, 86

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 3, 64, 81

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3, 5, 41, 45, 47, 64

[9] Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1492–1500, 2017. 3, 6, 65, 66, 67, 78, 79, 81, 87

[10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010. 3, 59

[11] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014. 3, 54

[12] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 2017. 6, 65, 66, 67, 81, 87

[13] Rangachari Anand, Kishan Mehrotra, Chilukuri K Mohan, and Sanjay Ranka. Efficient classification for multiclass problems using modular neural networks. *Neural Networks, IEEE Transactions on*, 6(1):117–124, 1995. 8

[14] Stefan Knerr, Léon Personnaz, and Gérard Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In *Neurocomputing*, pages 41–50. Springer, 1990. 8

[15] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *J. Artif. Int. Res.*, 2(1):263–286, January 1995. 8

[16] Miguel Ángel Bautista, Sergio Escalera, Xavier Baró, Petia Radeva, Jordi Vitriá, and Oriol Pujol. Minimal design of error-correcting output codes. *Pattern Recognition Letters*, 33(6):693 – 702, 2012. 9

[17] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141, 2004. 9

[18] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. In *Journal of Machine Learning Research*, pages 1453–1484, 2005. 9

[19] Erin L Allwein, Robert E Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *The Journal of Machine Learning Research*, 1:113–141, 2001. 9

[20] Mikel Galar, Alberto Fernndez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition*, 44(8):1761 – 1776, 2011. 9, 11

[21] Eibe Frank and Stefan Kramer. Ensembles of nested dichotomies for multi-class problems. In *Proceedings of the twenty-first international conference on Machine learning*, page 39. ACM, 2004. 9

[22] Jerome Friedman. Another approach to polychotomous classification. Technical report, Technical report, Department of Statistics, Stanford University, 1996. 10

[23] Bo Liu, Zhifeng Hao, and Eric CC Tsang. Nesting one-against-one algorithm based on svms for pattern classification. *Neural Networks, IEEE Transactions on*, 19(12):2044–2052, 2008. 10

[24] Eyke Hüllermeier and Stijn Vanderlooy. Combining predictions in pairwise classification: An optimal adaptive voting strategy and its relation to weighted voting. *Pattern Recognition*, 43(1):128–142, 2010. 10

[25] Trevor Hastie, Robert Tibshirani, et al. Classification by pairwise coupling. *The annals of statistics*, 26(2):451–471, 1998. 10, 14

[26] Ting-Fan Wu, Chih-Jen Lin, and Ruby C Weng. Probability estimates for multi-class classification by pairwise coupling. *The Journal of Machine Learning Research*, 5:975–1005, 2004. 10, 14

[27] John C Platt, Nello Cristianini, and John Shawe-Taylor. Large margin dags for multiclass classification. In *nips*, volume 12, pages 547–553, 1999. 10

[28] Eyke Hüllermeier and Klaus Brinker. Learning valued preference structures for solving classification problems. *Fuzzy Sets and Systems*, 159(18):2337–2352, 2008. 10

[29] Alberto Fernández, Marıa Calderón, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. Enhancing fuzzy rule based systems in multi-classification using pairwise coupling with preference relations. In *EUROFUSE Workshop Preference Modelling and Decision Analysis*, 2009. 10

[30] Ben Fei and Jinbai Liu. Binary tree of svm: a new fast multiclass training and classification algorithm. *Neural Networks, IEEE Transactions on*, 17(3):696–704, 2006. 10

[31] Johannes Fürnkranz, Eyke Hüllermeier, and Stijn Vanderlooy. Binary decomposition methods for multipartite ranking. In *Machine Learning and Knowledge Discovery in Databases*, pages 359–374. Springer, 2009. 11

[32] Mikel Galar, Alberto Fernndez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. Dynamic classifier selection for one-vs-one strategy: Avoiding non-competent classifiers. *Pattern Recognition*, 46(12):3412 – 3424, 2013. 12, 26

[33] Mikel Galar, Alberto Fernndez, Edurne Barrenechea, and Francisco Herrera. Drcw-ovo: Distance-based relative competence weighting combination for one-vs-one strategy in multi-class problems. *Pattern Recognition*, 48(1):28 – 42, 2015. 12, 23, 25, 26, 30, 31

[34] Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 694–699, New York, NY, USA, 2002. ACM. 13

[35] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pages 61–74. MIT Press, 1999. 13

[36] Michael J Best and Nilotpal Chakravarti. Active set algorithms for isotonic regression; a unifying framework. *Mathematical Programming*, 47(1-3):425–439, 1990. 13

[37] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 625–632. ACM, 2005. 13

[38] Eun Bae Kong and TG Diettrich. Probability estimation via error-correcting output coding. In *Int. Conf. of Artificial Inteligence and soft computing*. Citeseer, 1997. 14

[39] Berwin A Turlach. *Bandwidth selection in kernel density estimation: A review*. Université catholique de Louvain, 1993. 20

[40] Bernard W Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986. 21

[41] J. Alcal-Fdez, L. Snchez, S. Garca, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernndez, and F. Herrera. Keel: a software tool to assess evolutionary algorithms for data mining problems. *Soft Computing*, 13(3):307–318, 2009. 23

[42] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960. 26

[43] Jr. Hodges, J.L. and E.L. Lehmann. Rank methods for combination of independent experiments in analysis of variance. In Javier Rojo, editor, *Selected Works of E. L. Lehmann*, Selected Works in Probability and Statistics, pages 403–418. Springer US, 2012. 26

[44] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979. 26

[45] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83, 1945. 26

[46] V.I Morariu, E. Ahmed, V. Santhanam, D. Harwood, and L.S. Davis. Composite Discriminant Factor analysis. In *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, pages 564–571, March 2014. 33, 34, 39

[47] A. Quattoni and A. Torralba. Recognizing indoor scenes. pages 413–420, June 2009. 34, 35

[48] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013. 35

[49] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 35, 54

[50] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*, pages 512–519. IEEE, 2014. 35

[51] S. Sadanand and **J. J. Corso**. Action Bank Features for UCF50. `http://www.cse.buffalo.edu/~jcorso/r/actionbank/`. 35

[52] S. Sadanand and **J. J. Corso**. Action bank: A high-level representation of activity in video. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2012. 35, 36

[53] Kishore K Reddy and Mubarak Shah. Recognizing 50 human action categories of web videos. *Machine Vision and Applications*, 24(5):971–981, 2013. 35

[54] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. A maximum entropy framework for part-based texture and object recognition. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 832–838. IEEE, 2005. 36

[55] Svetlana Lazebnik, Cordelia Schmid, Jean Ponce, et al. Semi-local affine parts for object recognition. In *British Machine Vision Conference (BMVC'04)*, pages 779–788, 2004. 36

[56] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *Computer Vision–ECCV 2010*, pages 213–226. Springer, 2010. 36

[57] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 41

[58] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015. 41, 43, 47, 55

[59] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016. 41, 43

[60] Golnaz Ghiasi and Charless C Fowlkes. Laplacian pyramid reconstruction and refinement for semantic segmentation. In *European Conference on Computer Vision*, pages 519–534. Springer, 2016. 41

[61] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 447–456, 2015. 41, 45, 47

[62] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015. 41, 55

[63] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Deep end2end voxel2voxel prediction. *arXiv preprint arXiv:1511.06681*, 2015. 45

[64] Hansen F Chen, Peter N Belhumeur, and David W Jacobs. In search of illumination invariants. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pages 254–261. IEEE, 2000. 46

[65] Edwin H Land and John J McCann. Lightness and retinex theory. *JOSA*, 61(1):1–11, 1971. 46

[66] Ronen Basri and David W Jacobs. Lambertian reflectance and linear subspaces. *IEEE transactions on pattern analysis and machine intelligence*, 25(2):218–233, 2003. 46

[67] Yang Wang, Lei Zhang, Zicheng Liu, Gang Hua, Zhen Wen, Zhengyou Zhang, and Dimitris Samaras. Face relighting from a single image under arbitrary unknown lighting conditions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(11):1968–1984, 2009. 46

[68] Amr Almaddah, Sadi Vural, Yasushi Mae, Kenichi Ohara, and Tatsuo Arai. Face relighting using discriminative 2d spherical spaces for face recognition. *Machine Vision and Applications*, 25(4):845–857, 2014. 46

[69] Chia-Ping Chen and Chu-Song Chen. Lighting normalization with generic intrinsic illumination subspace for face recognition. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1089–1096. IEEE, 2005. 46

[70] James Burnstone and Hujun Yin. Eigenlights: Recovering illumination from face images. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 490–497. Springer, 2011. 46

[71] Lei Zhang, Meng Yang, and Xiangchu Feng. Sparse representation or collaborative representation: Which helps face recognition? In *2011 International Conference on Computer Vision*, pages 471–478. IEEE, 2011. 46

[72] Qiang Zhang and Baoxin Li. Discriminative k-svd for dictionary learning in face recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2691–2698. IEEE, 2010. 46

[73] Gee-Sern Hsu and Ding-Yu Lin. Face recognition using sparse representation with illumination normalization and component features. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2013 Asia-Pacific*, pages 1–5. IEEE, 2013. 46

[74] Long Ma, Chunheng Wang, Baihua Xiao, and Wen Zhou. Sparse representation for face recognition based on discriminative low-rank dictionary learning. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2586–2593. IEEE, 2012. 46

[75] Biao Wang, Weifeng Li, and Qingmin Liao. Illumination variation dictionary designing for single-sample face recognition via sparse representation. In *International Conference on Multimedia Modeling*, pages 436–445. Springer, 2013. 46

[76] Vishal M Patel, Tao Wu, Soma Biswas, P Jonathon Phillips, and Rama Chellappa. Illumination robust dictionary-based face recognition. In *2011 18th IEEE International Conference on Image Processing*, pages 777–780. IEEE, 2011. 46

[77] Yichuan Tang, Ruslan Salakhutdinov, and Geoffrey Hinton. Deep lambertian networks. *arXiv preprint arXiv:1206.6445*, 2012. 46

[78] Weisheng Dong, Xin Li, Lei Zhang, and Guangming Shi. Sparsity-based image denoising via dictionary learning and structural clustering. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 457–464. IEEE, 2011. 46, 58

[79] Daniel Zoran and Yair Weiss. From learning models of natural image patches to whole image restoration. In *2011 International Conference on Computer Vision*, pages 479–486. IEEE, 2011. 46, 58

[80] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095, 2007. 46, 58

[81] Marc Lebrun, Antoni Buades, and Jean-Michel Morel. A nonlocal bayesian image denoising algorithm. *SIAM Journal on Imaging Sciences*, 6(3):1665–1688, 2013. 46, 58

[82] Weisheng Dong, Lei Zhang, Guangming Shi, and Xin Li. Nonlocally centralized sparse representation for image restoration. *IEEE Transactions on Image Processing*, 22(4):1620–1630, 2013. 46, 58

[83] Shuhang Gu, Lei Zhang, Wangmeng Zuo, and Xiangchu Feng. Weighted nuclear norm minimization with application to image denoising. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2862–2869, 2014. 46, 58

[84] Forest Agostinelli, Michael R Anderson, and Honglak Lee. Adaptive multi-column deep neural networks with application to robust image denoising. In *Advances in Neural Information Processing Systems*, pages 1493–1501, 2013. 46

[85] Viren Jain and Sebastian Seung. Natural image denoising with convolutional networks. In *Advances in Neural Information Processing Systems*, pages 769–776, 2009. 46

[86] Harold C Burger, Christian J Schuler, and Stefan Harmeling. Image denoising: Can plain neural networks compete with bm3d? In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2392–2399. IEEE, 2012. 46, 58

[87] Junyuan Xie, Linli Xu, and Enhong Chen. Image denoising and inpainting with deep neural networks. In *Advances in Neural Information Processing Systems*, pages 341–349, 2012. 46

[88] Shuangteng Zhang and Ezzatollah Salari. Image denoising using a neural network based non-linear filter in wavelet domain. In *Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, volume 2, pages ii–989. IEEE, 2005. 46

[89] Raviteja Vemulapalli, Oncel Tuzel, and Ming-Yu Liu. Deep gaussian conditional random field network: A model-based deep network for discriminative denoising. *arXiv preprint arXiv:1511.04067*, 2015. 46, 58, 59, 61

[90] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep colorization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 415–423, 2015. 47

[91] Yuji Morimoto, Yuichi Taguchi, and Takeshi Naemura. Automatic colorization of grayscale images using multiple images on the web. In *SIGGRAPH'09: Posters*, page 32. ACM, 2009. 47

[92] Guillaume Charpiat, Matthias Hofmann, and Bernhard Schölkopf. Automatic image colorization via multimodal predictions. In *European conference on computer vision*, pages 126–139. Springer, 2008. 47

[93] Tomihisa Welsh, Michael Ashikhmin, and Klaus Mueller. Transferring color to greyscale images. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 277–280. ACM, 2002. 47

[94] Raj Kumar Gupta, Alex Yong-Sang Chia, Deepu Rajan, Ee Sin Ng, and Huang Zhiyong. Image colorization using similar images. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 369–378. ACM, 2012. 47

[95] Xiaopei Liu, Liang Wan, Yingge Qu, Tien-Tsin Wong, Stephen Lin, Chi-Sing Leung, and Pheng-Ann Heng. Intrinsic colorization. *ACM Transactions on Graphics (TOG)*, 27(5):152, 2008. 47

[96] Alex Yong-Sang Chia, Shaojie Zhuo, Raj Kumar Gupta, Yu-Wing Tai, Siu-Yeung Cho, Ping Tan, and Stephen Lin. Semantic colorization with internet images. In *ACM Transactions on Graphics (TOG)*, volume 30, page 156. ACM, 2011. 47

[97] Aditya Deshpande, Jason Rock, and David Forsyth. Learning large-scale automatic image colorization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 567–575, 2015. 47

[98] Ryan Dahl. http://tinyclouds.org/colorize/, 2016. 47

[99] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be color!: joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics (TOG)*, 35(4):110, 2016. 47

[100] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. *arXiv preprint arXiv:1603.06668*, 2016. 47

[101] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. *arXiv preprint arXiv:1603.08511*, 2016. 47, 48

[102] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010. 51, 53

[103] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 51, 53

[104] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 53

[105] Léon Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer, 2012. 53

[106] Yunjin Chen, Wei Yu, and Thomas Pock. On learning optimized reaction diffusion processes for effective image restoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5261–5269, 2015. 58

[107] Pablo Arbelaez, Charless Fowlkes, and David Martin. The berkeley segmentation dataset and benchmark. *see http://www. eecs. berkeley. edu/Research/Projects/CS/vision/bsds*, 2007. 59

[108] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. *arXiv preprint arXiv:1603.08511*, 2016. 59, 60, 62, 63

[109] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 60

[110] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. *arXiv preprint arXiv:1608.03665*, 2016. 63

[111] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006. 63

[112] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *arXiv preprint arXiv:1603.08155*, 2016. 63

[113] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014. 63

[114] Venkataraman Santhanam, Vlad I Morariu, and Larry S Davis. Generalized deep image to image regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5609–5619, 2017. 64

[115] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, Jen-Hao Rick Chang, et al. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 64, 65

[116] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017. 64

[117] Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks. In *Advances in Neural Information Processing Systems*, pages 4470–4478, 2017. 64

[118] Masoud Abdi and Saeid Nahavandi. Multi-residual networks. *CoRR, abs/1609.05672*, 8, 2016. 64, 68

[119] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932, 2014. 64

[120] Matus Telgarsky. benefits of depth in neural networks. In *Conference on Learning Theory*, pages 1517–1539, 2016. 64

[121] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015. 64

[122] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. 2017. 65

[123] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 65, 66, 78, 79

[124] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 66, 87

[125] Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Advances in Neural Information Processing Systems*, pages 550–558, 2016. 66, 68, 74

[126] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision*, pages 646–661. Springer, 2016. 68

[127] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648*, 2016. 68

[128] Klaus Greff, Rupesh K Srivastava, and Jürgen Schmidhuber. Highway and residual networks learn unrolled iterative estimation. *arXiv preprint arXiv:1612.07771*, 2016. 68

[129] David Balduzzi, Marcus Frean, Lennox Leary, JP Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. The shattered gradients problem: If resnets are the answer, then what is the question? *arXiv preprint arXiv:1702.08591*, 2017. 68

[130] A Emin Orhan. Skip connections as effective symmetry-breaking. *arXiv preprint arXiv:1701.09175*, 2017. 68

[131] A Emin Orhan and Xaq Pitkow. Skip connections eliminate singularities. *arXiv preprint arXiv:1701.09175*, 2017. 68

[132] Yangqing Jia. Learning semantic image representations at a large scale. 2014. 81

[133] Kumar Chellapilla, Sidd Puri, and Patrice Simard. High Performance Convolutional Neural Networks for Document Processing. In Guy Lorette, editor, *Tenth International Workshop on Frontiers in Handwriting Recognition*, La Baule (France), October 2006. Université de Rennes 1, Suvisoft. http://www.suvisoft.com. 81

[134] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, Berkeley, CA, USA, 2016. USENIX Association. 81

[135] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1874–1883, 2016. 81

[136] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017. 81

[137] R. Timofte, E. Agustsson, L. V. Gool, M. H. Yang, L. Zhang, B. Lim, S. Son, H. Kim, S. Nah, K. M. Lee, X. Wang, Y. Tian, K. Yu, Y. Zhang, S. Wu, C. Dong, L. Lin, Y. Qiao, C. C. Loy, W. Bae, J. Yoo, Y. Han, J. C. Ye, J. S. Choi, M. Kim, Y. Fan, J. Yu, W. Han, D. Liu, H. Yu, Z. Wang, H. Shi, X. Wang, T. S. Huang, Y. Chen, K. Zhang, W. Zuo, Z. Tang, L. Luo, S. Li, M. Fu, L. Cao, W. Heng, G. Bui, T. Le, Y. Duan, D. Tao, R. Wang, X. Lin, J. Pang, J. Xu, Y. Zhao, X. Xu, J. Pan, D. Sun, Y. Zhang, X. Song, Y. Dai, X. Qin, X. P. Huynh, T. Guo, H. S. Mousavi, T. H. Vu, V. Monga, C. Cruz, K. Egiazarian, V. Katkovnik, R. Mehta, A. K. Jain, A. Agarwalla, C. V. S. Praveen, R. Zhou, H. Wen, C. Zhu, Z. Xia, Z. Wang, and Q. Guo. Ntire 2017 challenge on single image super-resolution: Methods and results. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1110–1121, July 2017. 81

[138] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016. 81